Performance Analysis of Selected String Matching Algorithms based on Good Suffix and Bad Character Rule

G.L. Prajapati, PhD
Computer Engineering Dept.,
Institute of Engineering & Technology,
Devi Ahilya University, Indore, INDIA

Abhijeet Singh Rathore
Computer Engineering Dept.,
Institute of Engineering & Technology,
Devi Ahilya University, Indore, INDIA

Bhavana Tanwar
Computer Engineering Dept.,
Institute of Engineering & Technology,
Devi Ahilya University, Indore, INDIA

Surbhi Bhadviy
Computer Engineering Dept.,
Institute of Engineering & Technology,
Devi Ahilya University, Indore, INDIA

Tushar Jain
Computer Engineering Dept.,
Institute of Engineering & Technology,
Devi Ahilya University, Indore, INDIA

ABSTRACT
String matching is a problem where a pattern is to be searched within a text. In this paper, we study about selected string matching algorithms which compute shifts; based on good suffix rule and/or bad character rule or their variations. Algorithms are compared on the basis of their execution time for different data sets; those differ on patterns and alphabet sizes. Finally, we present a summary for the selection of these algorithms in different applications, based on the experimental results obtained.

General Terms
Pattern Matching, Algorithms, Theoretical Computer Science.

Keywords
Good Suffix Rule, Bad Character Rule, Boyer Moore Variations, String Matching Problem, Performance Analysis.

1. INTRODUCTION
String matching is an important problem which involves searching for a string of pattern within a text. A number of algorithms are suggested in literature, for this problem. Every algorithm has different applicability based on different data sets. Among these it is difficult to select an algorithm and use it for a specific application. The target is to simplify this selection process for some applications.

The paper is a study for performance analysis of six string matching algorithms, namely Boyer Moore[1][2], Turbo Boyer Moore[3], Zhu Takaoka[4] ,Smith[5], Horspool[6],and Quick Search (Sunday)[7].Initial steps involve proper understanding of the given algorithms .The algorithms are run in String Matching Research Tool (SMART)[8], and the results obtained have been recorded for a rigorous analysis. Here patterns of different size are generated from the text itself and then running time of all algorithms is recorded. The patterns under study vary from size of 2^0 to 2^{11}. Every pattern is run for 500 times for one algorithm and then the average time is considered as the result.

Different algorithms work differently on different pattern size, alphabet size or broadly speaking on various data sets. The objective is to suggest selection of appropriate algorithms for various data sets. In the following sections we discuss main results obtained during analysis.

2. DEFINITIONS AND NOTATIONS
2.1 Rules Used In Algorithms
Good Suffix:
Rule says if a mismatch occurs between the character ‘a’ of the pattern X and the character ‘b’ of the text Y during an attempt at position ‘j’ in Y. Then the already matched set of characters is considered as a suffix. The good-suffix shift consists in aligning this suffix with its rightmost occurrence in X that is preceded by a character different from character at position j. If there exists no such segment, the shift consists in aligning the longest set of characters in the suffix in Y with a matching prefix of X. Else shift the pattern ahead of suffix.

Bad character:
The bad-character shift consists in aligning the mismatched text character ‘b’ in Y with its rightmost occurrence in X. If ‘b’ does not occur in the pattern X, no occurrence of X in Y can include ‘b’, then the left end of the window is aligned with the character immediately after b.

2.2 Selected Algorithms
Boyer Moore Algorithm [1][2]: The algorithm scans the characters of the pattern from right to left beginning with the rightmost one. In case of a mismatch (or a complete match of the whole pattern) it uses two precomputed functions to shift the window to the right. These two shift functions are called the good-suffix shift (also called matching shift and the bad-character shift (also called the occurrence shift). Maximum of the two rules is taken and patterns is shifted accordingly.
Turbo Boyer Moore [3]: The Turbo-BM algorithm is an amelioration of the Boyer-Moore algorithm. It needs no extra preprocessing and requires only a constant extra space with respect to the original Boyer-Moore algorithm. It improves the worst-case complexity of Boyer-Moore algorithm.
Zhu Takoaka [4]: Zhu and Takaoka designed an algorithm which performs the shift by considering the bad-character shift for two consecutive text characters. It simply modifies the Bad character rule employed in Boyer Moore, to consider two characters in case of mismatch.

Smith [5]: As observed computing the shift with the text character just next the rightmost text character of the window gives sometimes shorter shift than using the rightmost text character of the window. This algorithm then takes the maximum between the two values.

Horspool [6]: Using the bad-character shift alone produces a very efficient algorithm in practice, when the alphabet is large compared with the length of the pattern. Horspool proposed to use only the bad-character shift of the rightmost character of the window to compute the shifts in the Boyer-Moore algorithm.

Quick Search [7]: The bad-character shift of the Boyer Moore algorithm is slightly modified to take into account the last character of the pattern, and then shift the pattern accordingly.

3. EXPERIMENTAL RESULTS

3.1 Machine Architecture

The tests were run on 2288.922 MHz CPU with 4289.84 BogoMIPs. Number of CPU(s) and socket is 1 having 1 Thread(s) per core and 1 Core(s) per socket. Byte Order of CPU used is Little Endian. Operating System: Ubuntu 11.0.1.

3.2 Context of observation:

1. To have a deterministic performance of an algorithm, the experimental section deals with performance of selected algorithms on various data sets.
2. In all 10 data sets have been chosen, which differ from each other based on alphabet size. Out of which 8 files are random texts of increasing alphabet size, one file covers English alphabet and one file is of Genome (DNA).
3. The size of alphabet in files vary from 2^1 to 2^8. Each algorithm is run on every file for patterns of length varying from 2^0 to 2^{11} and their time in milli seconds has been recorded.
4. Each Table records running time of the algorithms for different data sets.
5. Abbreviations used: BF: Brute Force, BM: Boyer Moore, HOR: Horspool, QS: Quick Search, TBM: Turbo Boyer Moore, ZT: Zhu Takoaka.

3.3 Observations based on alphabet size

In this section an analysis is presented based on alphabet size of text. The results and observations have been maintained in 10 different tables namely Table-1, Table-2, Table-3, Table-4, Table-5, Table-6, Table-7, Table-8, Table-9, and Table-10.

3.3.1 Alphabet Size = 2

When number of alphabet is 2, Table-1 shows experimental results. The observations made are, Zhu Takoaka comes out to be best for Alphabet size 2. For patterns of very small length, Smith proves to be better because Zhu Takoaka computes the bad character shift by considering two characters, thereby increasing its preprocessing time. Whereas original Bad character rule uses one mismatched character to compute the shift.

3.3.2 Alphabet Size = 4

When number of alphabet is 4, Table-2 shows experimental results. The observations made are, no other algorithm performs better than Zhu Takoaka. This has been verified from Table-10 containing results from genome file which too has an alphabet size of 4.

3.3.3 Alphabet Size = 8

When number of alphabet is 8, Table-3 shows experimental results. The observations made are, Zhu Takoaka again comes out to be efficient for alphabet size 8. Also, now as alphabet size increases preprocessing time also plays a role. Thus for short patterns Brute Force is more efficient as there is no preprocessing involved.

3.3.4 Alphabet Size = 16

When number of alphabet is 16, Table-4 shows experimental results. The observations made are, Brute Force is better for very short patterns. And Quick Search is better than Zhu Takoaka for shorter patterns due to extra preprocessing involved in Zhu Takoaka. As Zhu Takoaka modifies Bad character rule to take into account two characters. Overall performance wise Zhu Takoaka is better.

3.3.5 Alphabet Size = 32

When number of alphabet is 32, Table-5 shows experimental results. The observations made are, Zhu Takoaka and Quick Search are comparative for this file of alphabet size 32. But gradually performance of Zhu Takoaka suffers due to increase in preprocessing time. However Quick Search is better for this alphabet size.

3.3.6 Alphabet Size = 64

When number of alphabet is 64, Table-6 shows experimental results. The observations made are, analysis shows ranking of algorithms as QS>Smith>TBM. Brute Force now becomes better for more patterns of increasing length due to greater preprocessing time involved in other algorithms.

3.3.7 Alphabet Size = 128

When number of alphabet is 128, Table-7 shows experimental results. The observations made are, no one algorithms proves to be best. But Quick Search, Smith and Turbo Boyer Moore can be compared based on differences in timing involved.

3.3.8 Alphabet Size = 256

When number of alphabet is 256, Table-8 shows experimental results. The observations made are, Horspool and Quick Search compete for better performance. It was seen that performance of Horspool increases for short sized patterns and large alphabet size i.e. where there is significant difference between sizes of alphabet and pattern (but size of alphabet should be greater). Here Horspool comes to be more efficient in majority cases.

3.3.9 Alphabet Size = 26 (English Text)

When number of alphabet is 26, Table-9 shows experimental results. The observations made are, Zhu Takoaka is better and in file of alphabet size 32, Quick Search is better. Here as alphabet size is 26 i.e. intermediate of 16 and 32 so we can observe Quick Search creeping in and giving better performance in some cases. Gradually Quick Search completely overpowers in file of alphabet size 64 and further files of increasing alphabet size.

3.3.10 Alphabet Size = 4 (Genome File/DNA)

Table-10 verifies the results obtained on alphabet size 4 for Zhu Takoaka, as Genome/DNA file also has an alphabet size of 4.

Table-1: Alphabet Size 2

Pattern Size	1	2	4	8	16	32	64	128	256	512	1024	2048
BF	17.14	19.41	20.74	18.46	19.24	20.24	19.73	19.43	18.39	18.58	19.31	18.93
BM	19.38	16.33	11.88	8.84	6.92	5.14	4.07	2.84	3.94	2.86	2.27	2.27
HOR	19.85	17.28	18.58	15.11	16.13	18.58	20.03	18.44	19.12	15.78	17.94	20.68
QS	15.46	17.92	17.12	17.34	20.99	19.94	19.7	16.98	16.42	16.13	18.48	20.95
SMITH	13.58	12.85	15.07	12.13	13.71	13.68	15.83	14.11	14.77	12.78	13.01	15.61
TBM	18.74	15.07	12.93	9.41	6.65	3.75	1.95	3.48	2.79	2.55	2.49	1.88
ZT	18.69	14.76	10.28	7.48	5.24	4.16	3.6	2.42	1.4	1.89	1.64	1.79
Best Time	13.58	12.85	10.28	7.48	5.24	3.75	1.95	2.42	1.4	1.89	1.64	1.88
Best Algorithm	Smith	Smith	ZT	ZT	ZT	TBM	TBM	ZT	ZT	ZT	ZT	ZT

Table-2: Alphabet Size 4

Pattern Size	1	2	4	8	16	32	64	128	256	512	1024	2048
BF	11.15	13.85	13.65	14.45	12.25	12.88	11.7	12.5	12.97	12.59	11.67	12.85
BM	15.85	11.04	8.35	6.24	6.2	4.59	4.17	3.79	2.14	2.99	2.5	2.42
HOR	16.95	10.53	6.7	5.19	6.46	5.83	5.48	5.75	5.74	4.42	4.73	4.74
QS	11.59	9.99	6.38	5.19	6.14	4.58	5.68	4.75	5.12	7.06	4.33	4.75
SMITH	10.77	8.61	5.76	5.57	4.24	4.7	4.96	4.32	3.78	5.99	3.29	3.86
TBM	16.34	13.57	8.9	6.89	7.07	4.54	3.76	4.35	4.54	3.09	3.62	2.49
ZT	13.75	7.04	3.49	3.11	1.52	2.27	1.34	2.61	0.87	1.52	1.15	1.13
Best Time	10.77	7.04	3.49	3.11	1.52	2.27	1.34	2.61	0.87	1.52	1.15	1.13
Best Algorithm	Smith	ZT	ZT	ZT	ZT	ZT	ZT	ZT	ZT	ZT	ZT	ZT

Table-3: Alphabet Size 8

Pattern Size	1	2	4	8	16	32	64	128	256	512	1024	2048
BF	7.08	9.53	8.1	6.65	6.84	7.7	7.07	7.42	7	6.99	7.68	8.19
BM	13.02	7.03	3.8	2.66	2.48	2.2	2.45	2.71	2.47	2.52	2.37	1.63
HOR	10.14	5.43	2.55	2.17	2.82	2.5	2.41	1.44	2.41	1.94	1.59	2.53
QS	8.15	5.01	3.65	2.56	2.99	2.06	1.63	2.2	2.23	3.24	1.49	1.88
SMITH	10.19	5.55	3.06	1.65	2.5	1.74	1.59	2.32	1.7	1.69	1.43	1.99
TBM	14.05	9.88	5.85	3.95	3.17	2.44	3.42	2.22	2.16	2.02	2.14	2.55
ZT	11.89	4.54	2.48	2.13	0.96	0.77	0.61	0.56	0.76	1.14	0.92	1.52
Best Time	7.08	4.54	2.48	1.65	0.96	0.77	0.61	0.56	0.76	1.14	0.92	1.52
Best Algorithm	BF	ZT	ZT	Smith	ZT	ZT	ZT	ZT	ZT	ZT	ZT	ZT

Table-4: Alphabet Size 16

Pattern Size	**1**	**2**	**4**	**8**	**16**	**32**	**64**	**128**	**256**	**512**	**1024**	**2048**
BF	3.72	4.45	4.42	5.04	3.56	4.38	4.15	2.72	5.11	4.98	4.48	4.69
BM	10.47	3.84	2.81	1.76	3.11	1.16	2.16	0.84	1.41	1.43	0.81	1.25
HOR	9.98	3.79	2.48	1.77	0.83	1.36	1.34	1.28	0.84	0.61	1.47	1.5
QS	7.11	3.74	1.43	1.24	0.83	1.74	1.59	1.51	0.95	1.4	1.16	1.37
SMITH	6.41	3.17	1.87	1.64	2.12	1.6	1.4	0.9	0.6	1.35	1.01	1.3
TBM	12.49	4.58	4.06	1.89	1.66	0.94	1.4	1.62	1.58	0.8	1.8	1.43
ZT	9.18	5.37	1.77	1.81	0.97	0.85	1.55	0.4	0.45	0.65	0.52	0.71
Best Time	3.72	3.17	1.43	1.24	0.83	0.85	1.34	0.4	0.45	0.61	0.52	1.25
Best Algorithm	BF	Smith	QS	QS	QS	ZT	HOR	ZT	ZT	HOR	ZT	ZT

Table-5: Alphabet Size 32

Pattern Size	**1**	**2**	**4**	**8**	**16**	**32**	**64**	**128**	**256**	**512**	**1024**	**2048**
BF	4.07	3.63	4.07	3.24	3.05	3.03	2.88	2.45	2.31	3.53	3.69	3.06
BM	11.52	5.39	3.09	1.55	1.27	1.79	0.52	1.12	0.8	0.89	0.67	1
HOR	8.24	3.99	2.21	0.7	0.84	0.9	1.18	0.67	0.9	0.45	0.95	1.92
QS	7.6	2.61	1.08	1.55	0.5	0.95	0.44	0.49	0.71	0.45	0.61	0.67
SMITH	4.79	2.74	1.52	1.91	0.78	0.46	0.95	1.18	0.84	1.76	0.94	0.46
TBM	9.56	5.64	2.34	2.06	1.93	1.49	0.85	0.55	1.56	0.55	0.81	1.23
ZT	10.02	4.66	1.82	1.09	0.65	0.45	0.4	0.95	1.02	0.94	0.51	0.43
Best Time	4.07	2.61	1.08	0.7	0.5	0.45	0.4	0.49	0.71	0.45	0.51	0.43
Best Algorithm	BF	QS	QS	HOR	QS	ZT	ZT	QS	QS	QS	ZT	ZT

Table-6: Alphabet Size 64

Pattern Size	**1**	**2**	**4**	**8**	**16**	**32**	**64**	**128**	**256**	**512**	**1024**	**2048**
BF	2.29	2.12	1.94	3.23	2.41	2.58	2.42	2.07	3.11	3.34	2.68	2.21
BM	11.18	3.22	3.88	1.63	0.57	0.47	0.77	1.28	0.94	0.84	0.71	0.52
HOR	7.76	2.81	2.04	0.87	1.05	0.61	0.44	0.79	1.01	0.38	1.13	1.25
QS	2.49	2.62	1.66	0.65	0.66	0.61	0.83	0.37	0.43	1.48	1.22	0.81
SMITH	4.27	3.11	2.58	1.51	0.53	0.45	0.38	0.71	1.63	1.05	0.79	0.84
TBM	12.01	4.46	2.67	1.71	1	1.13	0.42	0.49	0.88	1.05	0.67	1.22
ZT	12.03	5.22	3.27	0.99	1.12	0.55	1.06	0.85	0.79	0.98	0.95	0.85
Best Time	2.29	2.12	1.66	0.65	0.53	0.45	0.38	0.37	0.43	0.38	0.67	0.52
Best Algorithm	BF	BF	QS	QS	Smith	Smith	Smith	QS	QS	HOR	TBM	BM

Table-7: Alphabet Size 128

Pattern Size	1	2	4	8	16	32	64	128	256	512	1024	2048
BF	1.75	1.58	1.67	2.02	3.41	1.55	2.28	2.08	2.32	3.09	2.35	2.97
BM	10.74	4.27	2.25	1.78	0.87	0.81	0.39	1.46	0.73	0.86	0.55	1.26
HOR	7.29	2.39	1.85	0.98	0.89	0.62	1	0.99	0.71	1.14	1.5	0.83
QS	5.06	2.67	1.31	0.96	1.26	0.84	0.42	0.96	0.73	0.4	1.19	0.53
SMITH	5.85	2.93	1.91	0.7	0.83	0.78	0.4	0.33	0.38	0.88	1.29	1.44
TBM	10.11	4.56	2.48	1.44	1	0.49	0.49	0.93	0.98	0.45	0.48	0.87
ZT	12.16	4.69	3.27	1.85	1.27	1.15	0.88	0.84	0.39	0.54	0.5	0.96
Best Time	1.75	1.58	1.31	0.7	0.83	0.49	0.39	0.33	0.38	0.4	0.48	0.53
Best Algorithm	BF	BF	QS	Smith	Smith	TBM	BM	Smith	Smith	QS	TBM	QS

Table-8: Alphabet Size 256

Pattern Size	1	2	4	8	16	32	64	128	256	512	1024	2048
BF	1.34	1.88	2.18	2.33	2.63	1.38	2.48	1.35	2.41	1.94	2.81	2.87
BM	8.86	4.67	2.83	1.1	1.56	0.8	0.39	0.97	0.76	0.47	1	1.22
HOR	8.21	4.08	2.41	0.58	0.82	0.38	0.31	0.9	0.4	0.43	0.53	0.38
QS	3.63	2.19	2.91	1.07	0.4	1.1	0.64	0.33	0.37	1.41	0.4	0.56
SMITH	4.1	5.03	4.13	1.37	1.05	0.39	0.34	1	0.41	1.22	0.37	1.38
TBM	10.92	5.45	2.72	1.41	1.07	0.97	0.44	0.42	0.75	0.47	0.93	0.42
ZT	14.2	7.38	4.78	2.72	0.85	0.76	0.93	0.99	0.4	1.13	0.31	0.68
Best Time	1.34	1.88	2.18	0.58	0.4	0.38	0.31	0.33	0.37	0.43	0.31	0.38
Best Algorithm	BF	BF	BF	HOR	QS	HOR	HOR	QS	QS	HOR	ZT	HOR

Table-9: Alphabet Size 26 (English Text)

Pattern Size	1	2	4	8	16	32	64	128	256	512	1024	2048
BF	3.57	5.51	6.43	4.48	5.21	5.08	4.93	4.68	4.46	4.74	4.45	4.65
BM	12.74	7.53	2.9	2.07	1.31	1.55	1.65	1.24	0.96	0.73	0.43	0.56
HOR	9.24	4.37	3.06	1.6	1.5	1.2	1.43	0.57	0.87	1.05	0.91	0.83
QS	5.91	5.15	2.31	1.73	0.75	0.83	1.12	1.08	1.12	1.38	0.4	0.54
SMITH	6.83	4.27	2.9	1.99	1.58	0.88	0.46	1.14	0.38	0.42	0.87	0.34
TBM	11.05	8.15	3.25	1.22	1.72	1.38	1.36	1.81	1.17	0.71	0.5	0.38
ZT	11	4.62	3.29	1.88	1.13	1.03	0.46	0.44	0.46	0.39	0.39	0.33
Best Time	3.57	4.27	2.31	1.22	0.75	0.83	0.46	0.44	0.38	0.39	0.39	0.33
Best Algorithm	BF	Smith	QS	TBM	QS	QS	ZT	ZT	Smith	ZT	ZT	ZT

Table-10: Alphabet Size 4 (Genome File/DNA)

Pattern Size	1	2	4	8	16	32	64	128	256	512	1024	2048
BF	4.35	6.95	4.87	5.6	4.96	4.57	5.6	6.57	4.97	5.98	4.84	6.83
BM	7.36	4.35	3.37	2.29	1.71	1.76	1.52	1.56	1.24	1.52	1	1.32
HOR	8.02	4.83	1.99	3.35	1.91	2.72	2.44	2.71	2.51	2.41	1.76	2.53
QS	3.61	3.71	2.72	1.54	2.38	2.54	1.83	2.5	1.77	2.35	1.81	2.09
SMITH	4.73	2.92	2.72	2.11	2.14	1.16	1.79	2.16	1.53	2.8	2.27	1.84
TBM	9.94	4.84	2.98	2.79	2.54	3.35	2.34	1.42	1.88	1.01	1.21	1.48
ZT	7.62	3.38	2.01	1.42	0.62	1.22	0.51	1.34	0.71	0.47	0.93	0.47
Best Time	3.61	2.92	1.99	1.42	0.62	1.16	0.51	1.34	0.71	0.47	0.93	0.47
Best Algorithm	QS	Smith	HOR	ZT	ZT	Smith	ZT	ZT	ZT	ZT	ZT	ZT

Table-11: General Trends observed for all selected algorithms

Boyer Moore									
Pattern Size	**1**	**2**	**4**	**8**	**16**	**32**	**64**	**128**	**256**
Alphabet Size 2	19.38	16.33	11.88	8.84	6.92	5.14	4.07	2.84	3.94
Alphabet Size 4	15.85	11.04	8.35	6.24	6.2	4.59	4.17	3.79	2.14
Alphabet Size 8	13.02	7.03	3.8	2.66	2.48	2.2	2.45	2.71	2.47
Alphabet Size 16	10.47	3.84	2.81	1.76	3.11	1.16	2.16	0.84	1.41
Alphabet Size 32	11.52	5.39	3.09	1.55	1.27	1.79	0.52	1.12	0.8
Alphabet Size 64	11.18	3.22	3.88	1.63	0.57	0.47	0.77	1.28	0.94
Alphabet Size 128	10.74	4.27	2.25	1.78	0.87	0.81	0.39	1.46	0.73
Alphabet Size 256	8.86	4.67	2.83	1.1	1.56	0.8	0.39	0.97	0.76

TurboBM									
Pattern Size	**1**	**2**	**4**	**8**	**16**	**32**	**64**	**128**	**256**
Alphabet Size 2	18.74	15.07	12.93	9.41	6.65	3.75	1.95	3.48	2.79
Alphabet Size 4	16.34	13.57	8.9	6.89	7.07	4.54	3.76	4.35	4.54
Alphabet Size 8	14.05	9.88	5.85	3.95	3.17	2.44	3.42	2.22	2.16
Alphabet Size 16	12.49	4.58	4.06	1.89	1.66	0.94	1.4	1.62	1.58
Alphabet Size 32	9.56	5.64	2.34	2.06	1.93	1.49	0.85	0.55	1.56
Alphabet Size 64	12.01	4.46	2.67	1.71	1	1.13	0.42	0.49	0.88
Alphabet Size 128	10.11	4.56	2.48	1.44	1	0.49	0.49	0.93	0.98
Alphabet Size 256	10.92	5.45	2.72	1.41	1.07	0.97	0.44	0.42	0.75

Smith									
Pattern Size	**1**	**2**	**4**	**8**	**16**	**32**	**64**	**128**	**256**
Alphabet Size 2	13.58	12.85	15.07	12.13	13.71	13.68	15.83	14.11	14.77
Alphabet Size 4	10.77	8.61	5.76	5.57	4.24	4.7	4.96	4.32	3.78
Alphabet Size 8	10.19	5.55	3.06	1.65	2.5	1.74	1.59	2.32	1.7
Alphabet Size 16	6.41	3.17	1.87	1.64	2.12	1.6	1.4	0.9	0.6
Alphabet Size 32	4.79	2.74	1.52	1.91	0.78	0.46	0.95	1.18	0.84
Alphabet Size 64	4.27	3.11	2.58	1.51	0.53	0.45	0.38	0.71	1.63
Alphabet Size 128	5.85	2.93	1.91	0.7	0.83	0.78	0.4	0.33	0.38
Alphabet Size 256	4.1	5.03	4.13	1.37	1.05	0.39	0.34	1	0.41

Zhu Takoaka									
Pattern Size	**1**	**2**	**4**	**8**	**16**	**32**	**64**	**128**	**256**
Alphabet Size 2	18.69	14.76	10.28	7.48	5.24	4.16	3.6	2.42	1.4
Alphabet Size 4	13.75	7.04	3.49	3.11	1.52	2.27	1.34	2.61	0.87
Alphabet Size 8	11.89	4.54	2.48	2.13	0.96	0.77	0.61	0.56	0.76
Alphabet Size 16	9.18	5.37	1.77	1.81	0.97	0.85	1.55	0.4	0.45
Alphabet Size 32	10.02	4.66	1.82	1.09	0.65	0.45	0.4	0.95	1.02
Alphabet Size 64	12.03	5.22	3.27	0.99	1.12	0.55	1.06	0.85	0.79
Alphabet Size 128	12.16	4.69	3.27	1.85	1.27	1.15	0.88	0.84	0.39
Alphabet Size 256	14.2	7.38	4.78	2.72	0.85	0.76	0.93	0.99	0.4

Horspool									
Pattern Size	**1**	**2**	**4**	**8**	**16**	**32**	**64**	**128**	**256**
Alphabet Size 2	19.85	17.28	18.58	15.11	16.13	18.58	20.03	18.44	19.12
Alphabet Size 4	16.95	10.53	6.7	5.19	6.46	5.83	5.48	5.75	5.74
Alphabet Size 8	10.14	5.43	2.55	2.17	2.82	2.5	2.41	1.44	2.41
Alphabet Size 16	9.98	3.79	2.48	1.77	0.83	1.36	1.34	1.28	0.84
Alphabet Size 32	8.24	3.99	2.21	0.7	0.84	0.9	1.18	0.67	0.9
Alphabet Size 64	7.76	2.81	2.04	0.87	1.05	0.61	0.44	0.79	1.01
Alphabet Size 128	7.29	2.39	1.85	0.98	0.89	0.62	1	0.99	0.71
Alphabet Size 256	8.21	4.08	2.41	0.58	0.82	0.38	0.31	0.9	0.4

Quick Search									
Pattern Size	**1**	**2**	**4**	**8**	**16**	**32**	**64**	**128**	**256**
Alphabet Size 2	15.46	17.92	17.12	17.34	20.99	19.94	19.7	16.98	16.42
Alphabet Size 4	11.59	9.99	6.38	5.19	6.14	4.58	5.68	4.75	5.12
Alphabet Size 8	8.15	5.01	3.65	2.56	2.99	2.06	1.63	2.2	2.23
Alphabet Size 16	7.11	3.74	1.43	1.24	0.83	1.74	1.59	1.51	0.95
Alphabet Size 32	7.6	2.61	1.08	1.55	0.5	0.95	0.44	0.49	0.71
Alphabet Size 64	2.49	2.62	1.66	0.65	0.66	0.61	0.83	0.37	0.43
Alphabet Size 128	5.06	2.67	1.31	0.96	1.26	0.84	0.42	0.96	0.73
Alphabet Size 256	3.63	2.19	2.91	1.07	0.4	1.1	0.64	0.33	0.37

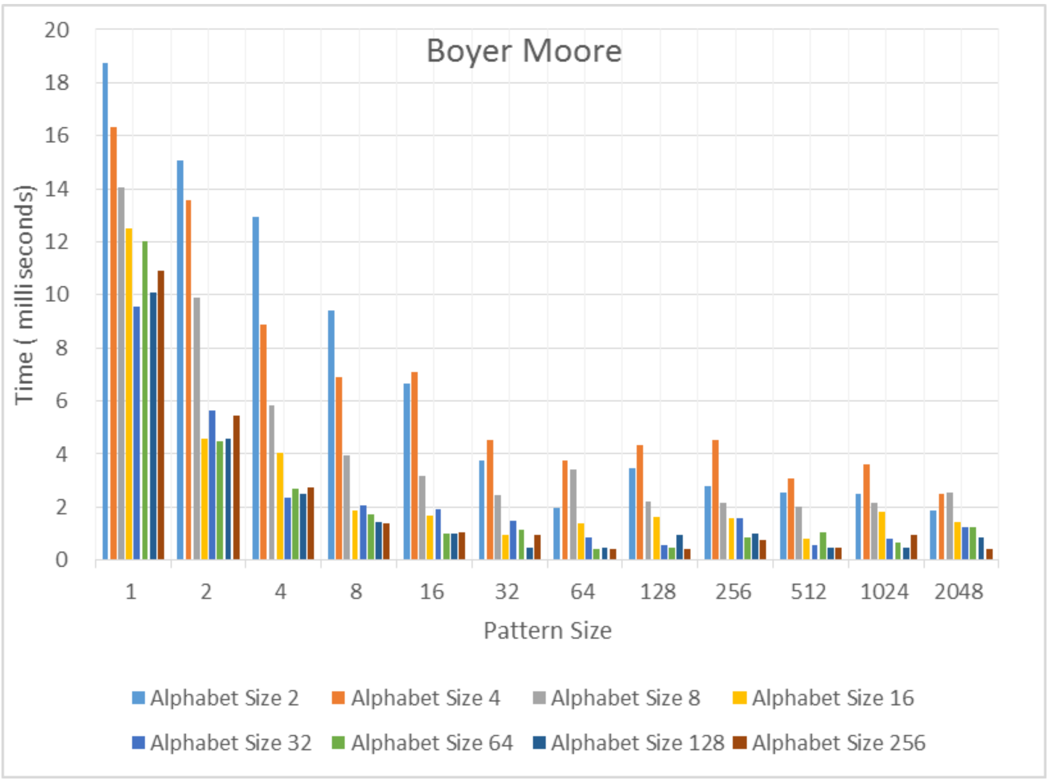

Figure-1

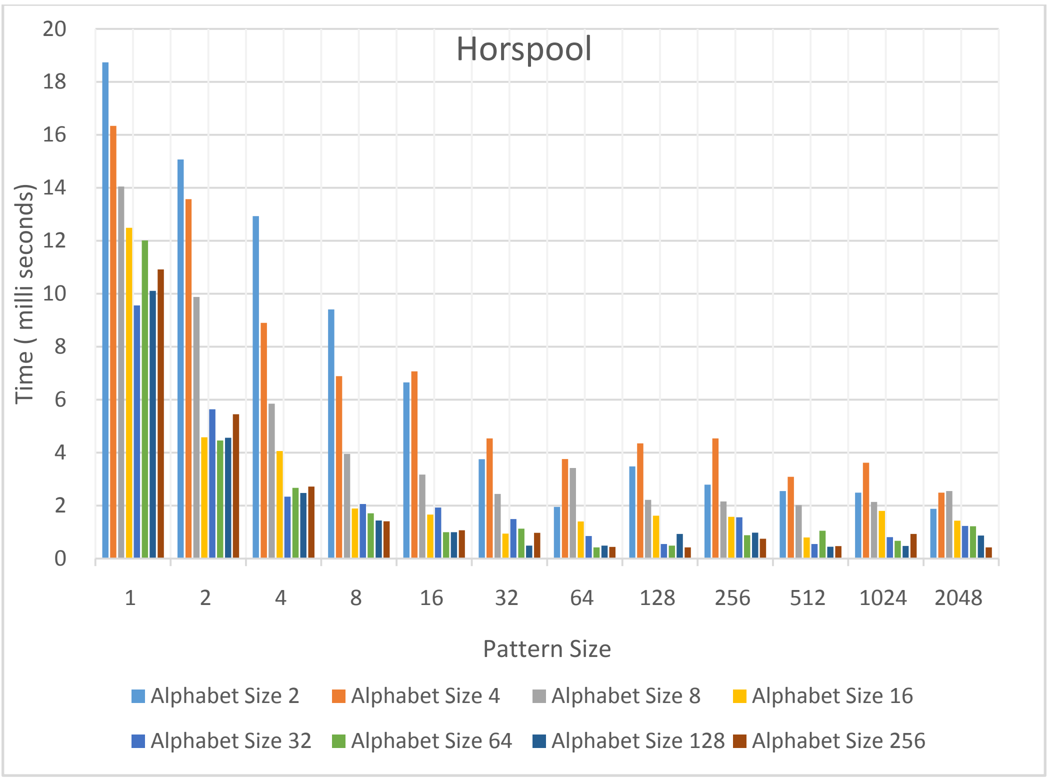

Figure-2

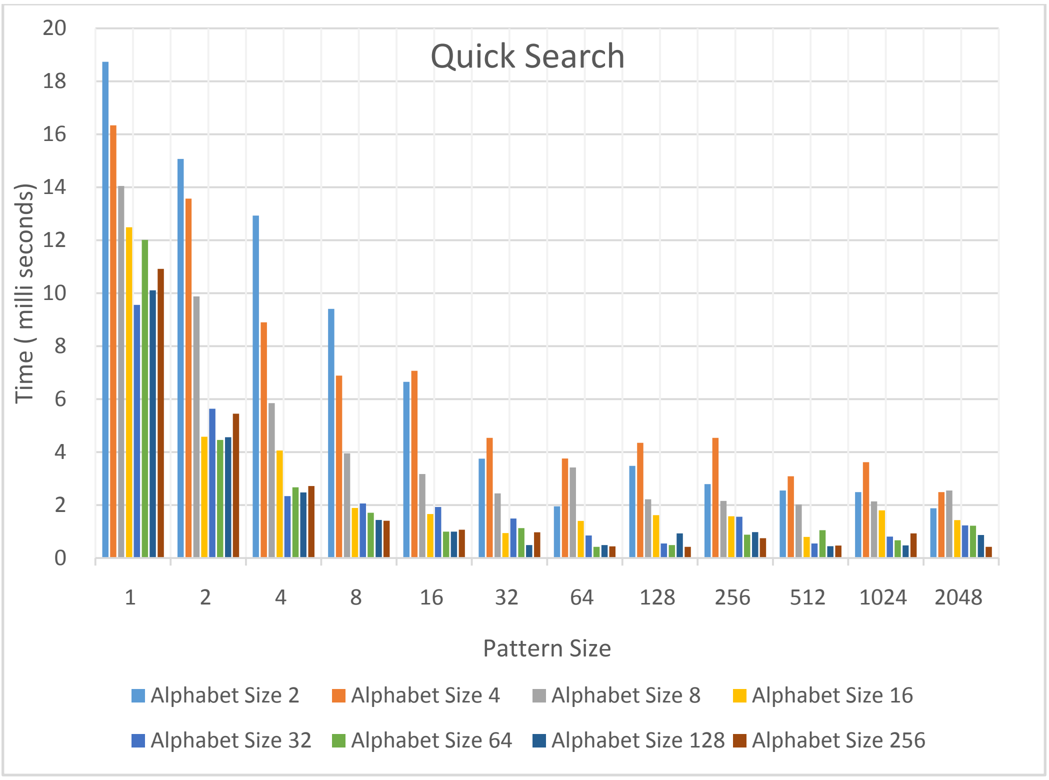

Figure-3

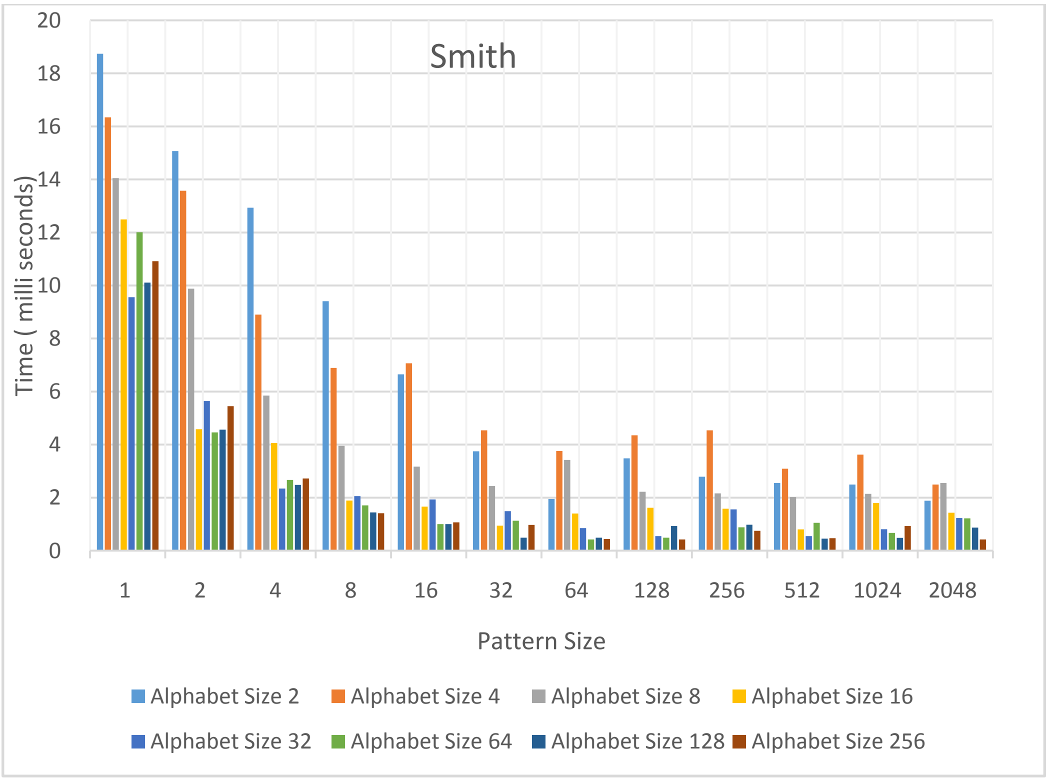

Figure-4

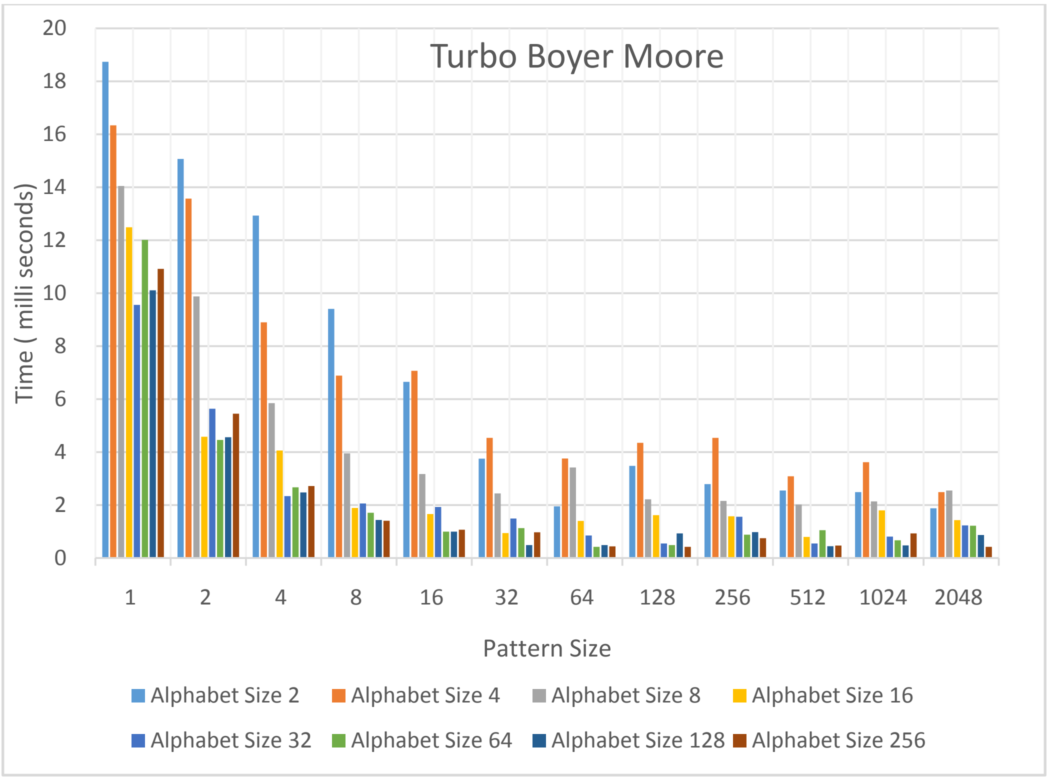

Figure-5

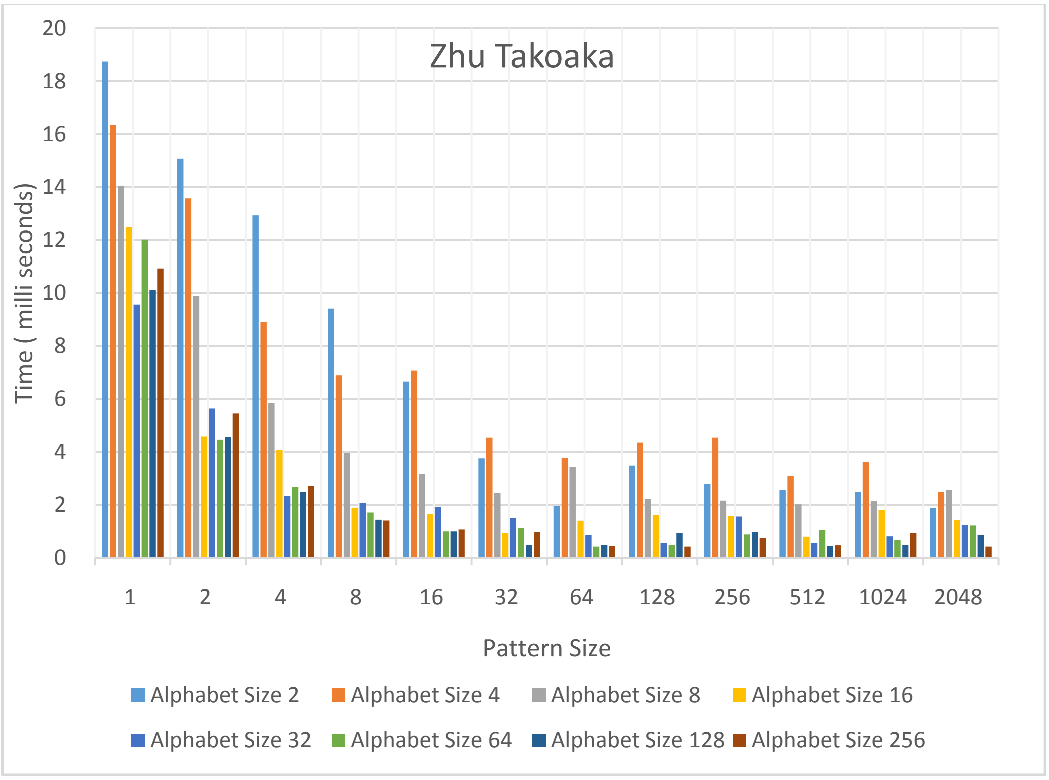

Figure-6

Table-12: Suggested Algorithms pertaining to specific pattern size and alphabet size

Pattern Size	1	2	4	8	16	32	64	128	256	512	1024	2048
Alphabet Size 2	Smith	Smith	ZT	ZT	ZT	TBM	TBM	ZT	ZT	ZT	ZT	ZT
Alphabet Size 4	Smith	ZT	ZT	ZT	ZT	ZT	ZT	ZT	ZT	ZT	ZT	ZT
Alphabet Size 8	BF	ZT	ZT	Smith	ZT	ZT	ZT	ZT	ZT	ZT	ZT	ZT
Alphabet Size 16	BF	Smith	QS	QS	QS	ZT	HOR	ZT	ZT	HOR	ZT	ZT
Alphabet Size 32	BF	QS	QS	HOR	QS	ZT	ZT	QS	QS	QS	ZT	ZT
Alphabet Size 64	BF	BF	QS	QS	Smith	Smith	Smith	QS	QS	HOR	TBM	BM
Alphabet Size 128	BF	BF	QS	Smith	Smith	TBM	BM	Smith	Smith	QS	TBM	QS
Alphabet Size 256	BF	BF	BF	HOR	QS	HOR	HOR	QS	QS	HOR	ZT	HOR

3.4 Graphical observations

Performance of the selected algorithms have also been illustrated via graphical data which aids in a comprehensive evaluation. The vertical axis represents execution time of algorithm in milli seconds and horizontal axis represents pattern size. Different colors in graphs are used to show time for specific alphabet sizes which have been mentioned in the legends. Fig-1, Fig-2, Fig-3, Fig-4, Fig-5, Fig-6 depict each of the selected algorithm.

3.5 General Trends Observed

Table-1 to Table-10 dealt with performance of individual algorithms corresponding to a particular alphabet size. However some general trends were also observed in all the selected algorithms. These trends have been depicted in Table-11. The observations were: Proceeding horizontally towards the right in the table the pattern size increases and execution time of all algorithms decreases. Proceeding vertically downwards the alphabet size increases and execution time of all selected algorithms decreases.

4. CONCLUSIONS

On the basis of experimental results obtained most efficient algorithms (with least execution time) pertaining to specific pattern size and alphabet size have been suggested under Table-12. This table presents a summary for selection of algorithms corresponding to various data sets. For example when pattern size is 16 and alphabet size is 32 then algorithm which has the least execution time is Quick Search (QS), and similarly algorithms can be selected for specific pattern and alphabet sizes.

Some conclusions regarding individual algorithms have also been drawn out. For short alphabet size (<16) Zhu Takoaka is better than others for pattern size of 2^0 to 2^{11}. As alphabet size increases Zhu Takoaka's performance begins to slow down due to increase in preprocessing time.

Brute Force begins to show better performance than other algorithms for short patterns in increasing alphabet size as no preprocessing is involved in Brute Force.

For string matching problems regarding English Text and Genome/DNA files Zhu Takoaka algorithm should be employed.

5. REFERENCES

[1] Boyer R.S., Moore J.S., 1977, a fast string searching algorithm. Communications of the ACM. 20:762-772.

[2] Aho, A.V., 1990, Algorithms for finding patterns in strings. in Handbook of Theoretical Computer Science, Volume A, Algorithms and complexity, J. van Leeuwen ed., Chapter 5, pp 255-300, Elsevier, Amsterdam.

[3] Crochemore, M., 1997. Off-line serial exact string searching, in Pattern Matching Algorithms, ed. A. Apostolico and Z. Galil, Chapter 1, pp 1-53, Oxford University Press.

[4] Zhu R.F., Takaoka T., 1987, on improving the average case of the Boyer-Moore string matching algorithm, Journal of Information Processing 10(3):173-177.

[5] Smith P.D., 1991, Experiments with a very fast substring search algorithm, Software Practice & Experience 21(10):1065-1074.

[6] Horspool R.N., 1980, Practical fast searching in strings, Software - Practice & Experience, 10(6):501-506.

[7] Sunday D.M., 1990, a very fast substring search algorithm, Communications of the ACM. 33(8):132-142

[8] The SMART tool used for execution of algorithms can be found at: http://www.dmi.unict.it/~faro/smart/.