12-1-1989

On the Average-Case Running Time of the Boyer-Moore Algorithm

Russell W. Quong
Purdue University

On the Average-Case Running Time of the Boyer-Moore Algorithm

Russell W. Quong

TR-EE 89-61
December 1989

School of Electrical Engineering
Purdue University
West Lafayette, Indiana 47907

On the Average-Case Running Time of the Boyer-Moore Algorithm

Russell W. Quong
School of Electrical Engineering
Purdue University
West Lafayette, IN

Abstract

The Boyer-Moore algorithm (BM) is a fast, compact algorithm for finding all occurrences of a pattern string in a text string. Previous papers have addressed the worst-case running time of BM, which occurs rarely in practice. In this paper, we derive an approximation to $\overline{\Phi(BM)}$, the average number of character probes made by BM. Let M = pattern length, N = text string length, α = the alphabet size, q = $1/\alpha$, and $\bar{q} = 1 - q$. By modeling BM as a probabilistic finite automaton, we show that

$$\overline{\Phi(BM)} \approx N \frac{q}{\bar{q}(1-\bar{q}^M)} \qquad \text{when } M \leq \alpha$$

and that

$$\overline{\Phi(BM)} \approx N \frac{q(1+q^2\bar{q}^2)}{\bar{q}^2 + \bar{q}(1-q\bar{q})\sum_{i=1}^{M}(1-\bar{q}^i)(1-e^{-q^i\bar{q}M})} \qquad \text{when } M > \alpha.$$

An immediate consequence is that $\overline{\Phi(BM)}$ is $O(N/\log_\alpha M)$, as $M \to \infty$. The above formulas match well with measured data.

Key Words: Algorithm, average-case complexity, pattern, pattern matching, string, string searching.

1 Introduction

String searching is a common, fundamental problem in computer science. String searching involves finding all occurrences of a *pattern* string in a *text* string, by examining or *probing* different text characters. For example, the pattern "or" occurs twice in the text "Boyer-Moore Algorithm". Let M = the length of the pattern and N = the length of the text string. The probability a random pattern character matches a random text character is q. Let $\bar{q} = 1 - q$. The effective alphabet size is $\alpha = 1/q$. If the pattern and text are uniformly distributed over the alphabet, then α = the size of the underlying alphabet.

In 1974, R. Boyer and J. Moore [2] and (independently) R.W. Gosper developed an algorithm, BM, that ran much faster than previously known algorithms. BM runs in "sublinear" time, because significantly fewer than N characters are examined on the average. The average number of probes made by BM, denoted $\overline{\Phi(BM)}$, decreases as M increases. In particular, when the length of the pattern is short compared to the

alphabet size, $\overline{\Phi(BM)}$ is roughly N/M. Because BM requires $O(1)$ time per probe, its expected running time is proportional to $\overline{\Phi(BM)}$. We ignore the cost of preprocessing the pattern.

Analysing the running time of BM is not easy, because BM uses the better of two heuristics to guide its behavior. Guibas and Odlysko [3] proved in the worst-case, BM makes $4N$ probes if the pattern is not found. Apostolico and Giancarlo [1] proved a worst-case bound of $2N$ for a Boyer-Moore variant that partially remembers which characters have been examined. Schaback [5] numerically computed the average number of probes for a simplified BM, when $M = O(\alpha)$.

In this paper, we derive two equations that approximate $\overline{\Phi(BM)}$ for short and long patterns respectively. Figure 1 shows quantitative behavior of the two equations. Empirically, we find that the crossover point between the equations occurs when M $\approx$ 2 α.

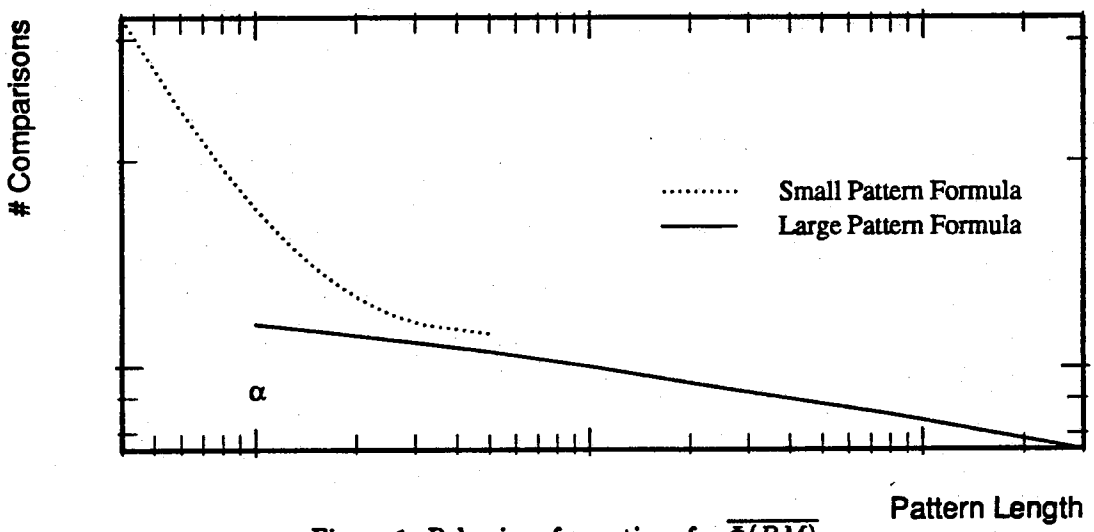

Figure 1: Behavior of equations for $\overline{\Phi(BM)}$.

First, we informally describe BM by example and summarise our results. Then, we model BM as a probalistic finite automaton, and derive state probabilities and expected slides. Assembling the pieces gives the desired result. Finally, we compare our results with measurements on random patterns and strings.

2 An overview of BM

BM [2] compares the pattern starting at the end and progresses to the front. On a mismatch, BM slides the pattern to the right realigning the pattern based on either (1) the text character just seen or (2) the current position within the pattern. BM resumes the search at the end of the new position.

The following example illustrates the sequence of probes and pattern alignments when searching for the pattern "abxabyab". A "*" represents an unseen text character. Note that only a fraction of the text is

probed.

```
probe                    1     4 3  2   6 5          10 9 8 7
text        * * * *  * * * x  * * b a  b * b b  * * * *  y y a b  * * * *
Alignment 1 a b x a  b y a b
Alignment 2            a b x  a b y a  b
Alignment 3                   a b x a  b y a b
Alignment 4                                   a b x a  b y a b
...                                                       a b  x a b y
```

Before searching the text, BM processes the pattern creating two tables, Δ_1 and Δ_2, which it consults on a mismatch. For a character c, $\Delta_1[c]$ = the rightmost occurrence of c in the pattern. Let m be the number of matching characters seen so far starting from the end of the pattern. $\Delta_2[m]$ = the rightmost plausible alignment, given that the last m characters of the pattern match, but the $m+1^{th}$ character does not match.

We show that if $M \leq \alpha$, the sublinearity of BM is due almost exclusively to the Δ_1 table, so that

$$\overline{\Phi(BM)} \approx N\frac{q}{\bar{q}(1-\bar{q}^M)} \approx \frac{N}{M}(1-\frac{(M+1)q}{2}).$$

When $M > \alpha$, the Δ_2 table becomes increasingly important, so that

$$\overline{\Phi(BM)} \approx N\frac{q(1+q^2\bar{q}^2)}{\bar{q}^2+\bar{q}(1-q\bar{q})\sum_{i=1}^{M}(1-\bar{q}^i)(1-e^{-q^i\bar{q}M})}.$$

We now examine the above search in detail to illustrate the use of the Δ_1 and Δ_2 tables.

Case 1) Use of the Δ_1 table. Here, $m = 0$ and the first probe mismatches on text character c. Align c with its rightmost occurrence in the pattern via $\Delta_1[c]$. In this example, c = the "x" at 8.

```
index   1 2 3 4 5 6 7 8 9 10 11
text    * * * * * * * x * * *
                      ↓
pattern a b x a b y a b          slide = 5, anchor = 0
                      ×
next                  a b x a b ...
```

Case 2) Slide into the middle via Δ_2.

Next, BM finds the "b" at 13 and then the "a" at 12. Because $m = 2$, on finding the "b" at 11, BM knows that "ab" has already been found. The current text "bab" does not match the rightmost "ab" pattern substring ("yab"), but the text might match the next "ab" substring which is different ("xab"). Note that Δ_2 ignores the character from the last probe, the "b" at 11, as it is indexed solely by m.

```
index    5  6  7  8  9  10 11 12 13 14 15
text     *  *  *  x  *  *  b  a  b  *  *
                          ↓
pattern     a  b  x  a  b  y  a  b        slide = 4, anchor = 6 (x)
                          ×  ✓  ✓
next                   a  b  x  a  b  y  ...
```

Case 3) Slide entirely past the current pattern via Δ_2.

Next, BM finds the matching "b" at 16, setting $m = 1$. The next probe finds the mismatching "b" at 15. The Δ_2 table gives a slide of 8, because in this pattern, every "b" is preceded by an "a". If the text does not match the "ab" at the end, no other "b" in the pattern can match.

```
index    8  9  10 11 12 13 14 15 16 17 18
text     x  *  *  b  a  b  *  b  b  *  *
                                ↓
pattern     a  b  x  a  b  y  a  b        slide = 8, anchor = 4,5 (a,b)
                                ×  ✓
next                                   a  b  ...
```

Case 4) Slide almost entirely past the current pattern, via a partial alignment via Δ_2.

BM finds "ayab" at 21–24. The "a" at 21 mismatches, with $m = 3$ and $\Delta_2[m] = 6$. The current text "ayab" does not match the current "yab" pattern substring ("byab"). No other "yab" substring exists in the pattern, but the text forms a *partial alignment* with the "ab" at the beginning of the pattern.

```
index    15 16 17 18 19 20 21 22 23 24 25
text     b  b  *  *  *  *  a  y  a  b  *
                                ×  ✓  ✓  ✓
pattern        a  b  x  a  b  y  a  b     slide = 6, anchor = 0
                                      a  b  ...
next
```

The *anchor*, A, is the set of previous matching probes that caused the most recent shift. The anchor is underlined above. The indices in A are measured from the end of the pattern with 1 being the last character in the pattern. For example, in case 2, A is $\{6\}$; in case 3, A is $\{4,5\}$. We set $A = 0$ when there is no anchor, as in cases 1 and 4. The *offending index* is pattern index of the mismatch. For example, the offending index is 3 in case 2.

On a mismatch, BM takes the maximum slide of the two tables. When $m = 0$ (case 1), Δ_1 always dictates the slide amount. For $m = 1$, each tables contributes the maximum about half the time. For $m \geq 2$, Δ_2 almost always has the greater slide amount, because it aligns m characters, whereas Δ_1 aligns just one character.

Simplification 1 *For $m \geq 2$, we assume Δ_2 always gives the slide.*

3 Modeling BM

We view BM as a probablistic finite automaton with transitions occurring on each probe. BM converges to a steady state probabilistically as the search progresses. We determine the probability and expected slide for each state. To search through the text, BM must slide the pattern past the text, so that asymptotically, the average number of probes is the text length divided by the expected slide over all states.

Formally, we model BM as an aperiodic, irreducible Markov chain. We derive the transition matrix, and then solve for the stationary probability vector. This type of Markov chain is guaranteed to converge to a stationary state over time.

The state $\xi_{A:m}$ means that m text characters have matched so far, with anchor A. For example, before case 1, we are in the *start state*, $\xi_{0:0}$. Immediately after the shift in case 1 above, we are in state $\xi_{6:0}$. In case 2, before the mismatch, we move through states $\xi_{6:0}$, $\xi_{6:1}$, and $\xi_{6:2}$. After sliding, we start case 3 in state $\xi_{5,4:0}$.

Simplification 2 *To simplify the analysis, we restrict A to a single index, a.* This assumption is reasonable except for $M \gg \alpha^2$.

The anchor holds future probes that are guaranteed to match if we get to them. If we match past the anchor, set $A = a = 0$. For example, on successive matches, BM might progress through states $\xi_{2:0} \rightarrow \xi_{2:1} \rightarrow \xi_{0:2} \rightarrow \xi_{0:3}$. Thus, either $a > m$ or $a = 0$. Note, when in state $\xi_{m+1:m}$, the next probe must match giving $\xi_{0:m+1}$. Also, there is no state $\xi_{1:0}$.

Our analysis consists of calculating

1. the probabilistic transition diagram,
2. $\Pr(\xi)$ = the steady-state probability of each state ξ, and
3. $\overline{s(\xi)}$ = the average slide when in state ξ on the next probe.

Once this has been done, the average slide per probe is

$$\overline{\text{slide}} = \frac{\sum \Pr(\xi)\overline{s(\xi)}}{\sum \Pr(\xi)}, \tag{1}$$

with both sums taken over all states. $\overline{\Phi(BM)}$ is asymptotic to $N/\overline{\text{slide}}$ for large N.

Let $\xi_{a*:m}$ be the set $\{\xi_{0:m}, \xi_{m+2:m}, \xi_{m+3:m}, \cdots, \xi_{M:m}\}$. The "*" indicates the next probe has not been seen previously. For all $\xi_1, \xi_2 \in \xi_{a*:m}$, $\overline{s(\xi_1)} = \overline{s(\xi_2)} = \overline{s(\xi_{a*:m})}$. Similarly, $\Pr(\xi_{a*:m}) = \sum_{\xi \in \xi_{a*:m}} \Pr(\xi)$. We derive the state probabilities from the following probabilistic transition table.

$\xi_{a:0}$	$\rightarrow$	$q\xi_{a:1} + \sum_{i=2}^{M} \bar{q}q^{i-1}\xi_{i:0} + \bar{q}^M\xi_{0:0}$	$\xi_{a:0} \in \xi_{a*:0}$	*Case* 1 (2)
$\xi_{m+1:m}$	$\rightarrow$	$\xi_{0:m+1}$	$1 \leq m \leq M-1$	
$\xi_{a:m}$	$\rightarrow$	$q\xi_{a:m+1} + \bar{q}\xi_{0:0}$	$\xi_{a:m} \in \xi_{a*:m}$	*Case* 3 (3)

We use $\sum$-notation to represent several states in the obvious way. For example, $\xi_{0:0} \rightarrow \sum_{i=2}^{M} \bar{q}q^{i-1}\xi_{i:0}$ means that BM goes from $\xi_{0:0}$ to $\xi_{2:0}$ with probability $\bar{q}q$, from $\xi_{0:0}$ to $\xi_{3:0}$ with probability $\bar{q}q^2$, and so on. We have used Simplifications 3 and 4 to simplify Equation 3. These assumptions affect $\Pr(\xi)$, but not $\overline{s(\xi)}$.

Simplification 3 *For $m \geq 2$, on a mismatch, we get a slide of M. We ignore the possibility of intermediate slides, namely cases 2 and 4.* This assumption follows directly from Simplification 2 which limits the anchor to a single index.

Simplification 4 *For $m = 1$, we have assumed that Δ_2 gives the slide, and on a mismatch, we enter state $\xi_{0:0}$.*

For convenience, we assign $\Pr(\xi_{0:0}) = 1$, as Equation 1 ignores constant factors in $\Pr(\xi)$. Let $\theta = \Pr(\xi_{a*:0})$. Note that immediately after a mismatch, BM is in $\xi_{a*:0}$. Collecting similar terms on the right side of the above equations gives $\Pr(\xi)$.

$$\Pr(\xi_{0:0}) = 1$$

$$\Pr(\xi_{a:0}) = q\bar{q}^{a-1}(\theta) \qquad 2 \leq a \leq M \qquad (4)$$

$$\Pr(\xi_{a:m}) = q^m \Pr(\xi_{a:0}) \qquad 1 \leq m < a \leq M \qquad (5)$$

$$\Pr(\xi_{0:m}) = q\Pr(\xi_{0:m-1}) + \Pr(\xi_{m:m-1}) \qquad 2 \leq m \leq M$$

Summing Equation 4 for $2 \leq a \leq$ M, gives $\theta - 1 = \theta(\bar{q} - \bar{q}^M)$, or $\theta = 1/(q + \bar{q}^M)$. Table 1 summarizes the state probabilities.

4 The expected slide

For a given ξ, the expected slide $\overline{s(\xi)}$ is the sum of possible slides, s, resulting from the next probe, weighted by their respective probabilities, $p(s)$. Thus, $\overline{s(\xi)} = \sum p(s)[s]$. Let c be the character found on the next probe.

State	*Probability*	
$\xi_{0:0}$	1	
$\xi_{a:m}$	$\theta q^{m+1}\bar{q}^{a-1}$	$m < a \leq M$
$\xi_{0:m}$	$q^m(1+\theta q^{-1}(\bar{q}-\bar{q}^m))$	
$\xi_{a*:m}$	$\theta q^{m-1}(1-q\bar{q})(1-\bar{q}^m)$.	

Table 1: State Probabilities

4.1 The slide from Δ_1

For $m = 0$, table Δ_1 dictates the slide amount. The probability that the pattern and text match is q, in which case the slide is 0. For a slide of i, where $1 \leq i \leq M-1$, the previous $i-1$ pattern characters must not match c, and the ith character must match c for a probability of $\bar{q}^{i-1}q$. If the pattern does not contain c, the slide is M.

$$\begin{array}{llll} \overline{s(\xi_{a*:0})} = & q[0] & + \sum_{i=1}^{M-1} q\bar{q}^i[i] & + \bar{q}^{M-1}[M]. \\ \textit{Slide} & 0 & i & M \end{array}$$

We have put the slide value in braces. Simplifying gives

$$\overline{s(\xi_{a*:0})} = \frac{\bar{q}(1-\bar{q}^M)}{q}. \tag{6}$$

4.2 The slide from Δ_2

For $m \geq 1$, we assume table $\Delta_2[m]$ dictates the amount of slide from Simplifications 1 and 4. Thus, $\overline{s(\xi_{a*:m})} = \Delta_2[m]$, which is important when $M \gg \alpha$. We summarize $p(s)$ and s, assuming a mismatch.

p(s)	Slide	Conditions	Case
$q^m\bar{q}(1-q^m\bar{q})^{s-1}$	s	$1 \leq s \leq M-m-1$	2
$q^m(1-q^m\bar{q})^{M-m-1}$	$M-m$		$2^\dagger(seetext)$
$(1-q^m\bar{q})^{M-m-1}\prod_{j=m+1-s}^{m}(1-q^j)$	$M-m+s$	$1 \leq s \leq m-1$	4
$(1-q^m\bar{q})^{M-m-1}\prod_{j=1}^{m}(1-q^j)$	M		3

The value of $p(s)$ is the probability that lesser slides do not align and slide s does align. In case 2, the probability for slide s is q^m (align m characters) times $\bar{q}$ (the character just before this alignment is different

from the character at the offending index) times $(1-q^m\bar{q})^{s-1}$ (the s-1 previous slides did not align). Case 2† occurs when the last m characters align with the beginning m characters of the pattern, eliminating the $\bar{q}$ factor from the offending index. Case 4 results in a partial alignment. The product term rules out previous partial alignments. Case 3 results when none of the previous possible alignments match.

Summing everything gives $\overline{S}$, the expected slide on a mismatch. A match occurs with probability q, in which case the slide is 0. Thus, $\overline{s(\xi_{a*:m})} = \bar{q}\overline{S}$.

$$\begin{aligned}\overline{S} &= q^m \sum_{i=1}^{M-m-1} (1-q^m\bar{q})^{i-1}[i] + (1-q^m\bar{q})^{M-m-1} q^m [M-m] + \\ &(1-q^m\bar{q})^{M-m-1} \sum_{i=1}^{m-1} \Big(q^{m-i} \prod_{j=m+1-i}^{m} (1-q^j)\Big)[M-m+i] + \\ &(1-q^m\bar{q})^{M-m-1} \Big(\prod_{j=1}^{m} (1-q^j)\Big)[M].\end{aligned}$$

For $x \ll 1$, $(1-x)^M \approx e^{-xM}$. Assuming $x = q^m\bar{q} \ll 1$, and $m \ll M$, then $(1-q^m\bar{q})^M \approx \exp(-xM)$. After simplifying and applying the approximation, the first two terms become $x^{-1}(1-\exp(-xM)) - M\exp(-xM)$. Judicious simplification of the last two terms yields $M\exp(-xM)$. Thus,

$$\overline{s(\xi_{a*:m})} = \bar{q}\overline{S} \approx \bar{q}\,\frac{1}{q^m\bar{q}}\big(1-\exp(-q^m\bar{q}M)\big). \tag{7}$$

Finally, $\overline{s(\xi_{m+1:m})} = 0$, because the next probe always matches.

5 Putting it together

All that remains is to apply Equation 1. We consider two cases.

When $M \le \alpha$, the dominant set of states is $\xi_{a*:0}$. We ignore the remaining states, greatly simplifying matters, because $\overline{\text{slide}} = \overline{s(\xi_{a*:0})}$. Thus, the "small M" approximation is

$$\overline{\Phi(BM)} \approx N\frac{q}{\bar{q}(1-\bar{q}^M)} \qquad \text{when } \mathrm{M} \le \alpha. \tag{8}$$

When $M \gg \alpha$, all the states are important. The sum of the probabilities $\sum \Pr(\xi) = \Pr(\xi_{a*:0}) + \sum_{m=1}^{M} \Pr(\xi_{a*:m}) + \sum_{m=1}^{M-1} \Pr(\xi_{m+1:m})$. We throw out all terms of order $O(q^M)$ and $O(\bar{q}^M)$, which gives $\sum \Pr(\xi) \approx \theta(1-q\bar{q}+q^2\bar{q}^2)/(\bar{q}(1-q\bar{q}))$. Using $(1-x+x^2)/(1-x) \approx 1+x^2$ with $x = q\bar{q}$ yields

$$\sum \Pr(\xi) \approx \frac{\theta(1+q^2\bar{q}^2)}{\bar{q}}. \tag{9}$$

In determining the numerator of Equation 1, the states $\xi_{m+1:m}$ contribute nothing because $\overline{s(\xi_{m+1:m})} = 0$. Thus, $\sum \Pr(\xi)\overline{s(\xi)} = \Pr(\xi_{a*:0})\overline{s(\xi_{a*:0})} + \sum_{m=1}^{M} \Pr(\xi_{a*:m})\overline{s(\xi_{a*:m})}$. Plugging in values from Table 1 and Equations 6 and 7 gives

$$\sum \Pr(\xi)\overline{s(\xi)} = \theta\left[\overline{s(\xi_{a*:0})}\right] + \sum_{m=1}^{M}\left[q^{m-1}\theta(1-q\bar{q})(1-\bar{q}^m)\right]\left[\bar{q}\frac{1}{q^m\bar{q}}(1-\exp(-q^m\bar{q}M))\right].$$

Dividing the weighted slide by Equation 9 gives $\overline{\text{slide}}$. Thus, the "large M" approximation is

$$\overline{\Phi(BM)} = N/\overline{\text{slide}} \approx N\frac{q(1+q^2\bar{q}^2)}{\bar{q}^2 + \bar{q}(1-q\bar{q})\sum_{i=1}^{M}(1-\bar{q}^i)(1-\exp(-q^i\bar{q}M))} \qquad \text{when } M > \alpha. \qquad (10)$$

The sum term behaves differently depending on $\bar{q}$ and M, and we were unable to simplify it. The $(1-\exp(-q^i\bar{q}M))$ factor quickly approaches zero once $i > \log_\alpha M$, so that in practice, we need only sum to $i = \log_\alpha 100M$.

An immediate result from Equation 10 is that $\overline{\Phi(BM)}$ is $O(N/\log_\alpha M)$ as $M \to \infty$. Let $L = \log_\alpha M$. The term $(1-\exp(-q^i\bar{q}M))$ is greater than $(1-e^{-1})$ for all $i \le L$. The quantity $\bar{q}^i = (1-1/\alpha)^i < e^{-i/\alpha}$. Thus, $(1-\bar{q}^i) \ge (1-e^{-1})$ for $i > \alpha$. As $M \to \infty$, $\sum(1-\bar{q}^i)(1-\exp(-q^i\bar{q}M)) > \sum_{i=\alpha}^{L}(1-e^{-1})(1-e^{-1})$ which is $O(\log_\alpha M)$.

6 Empirical Verification

To check the validity of our derivation, we compared our approximations against measured values of $\overline{\Phi(BM)}$. We averaged the data over many searches, using random pattern and text strings for various values of α and M. Our approximations agree quite closely with the measured data. The error term is [(predicted − measured)/measured] expressed as a percentage.

To test Equation 8 for small M, we averaged at least 100 searches for each value of M on a 50,000 character text file, for α = (4, 10, 26, 70). Figure 2 shows that the predicted slide values from Equation 8 are almost identical to the measured results except for $\alpha = 4$. For large α(26, 70), the predicted and measured results are virtually indistinguishable. Figure 3 shows that the error from Equation 8 decreases as α increases. For $\alpha \ge 10$, the error is always less 4%, with a typically value of 1%. For α= 4, the error was between 9% and 20%, although $M = 20$ is not really a small pattern. In all cases, Equation 8 under estimates the slide value, because it ignores the helpful effects from the Δ_2 table.

To test Equation 10, we averaged 100 searches for each value of M on a text file with at least 1,000,000 characters for α = (3, 4, 10, 20), and for M between 27 and 3000. Figure 4 shows that the predicted slide values match well with the measured results in all cases when $M > 2\alpha$. For very large M, Equation 10 becomes less accurate as shown when $\alpha = 10$ and $M > 1000$. Figure 5 shows the error from Equation 10 is less than 5% in most cases.

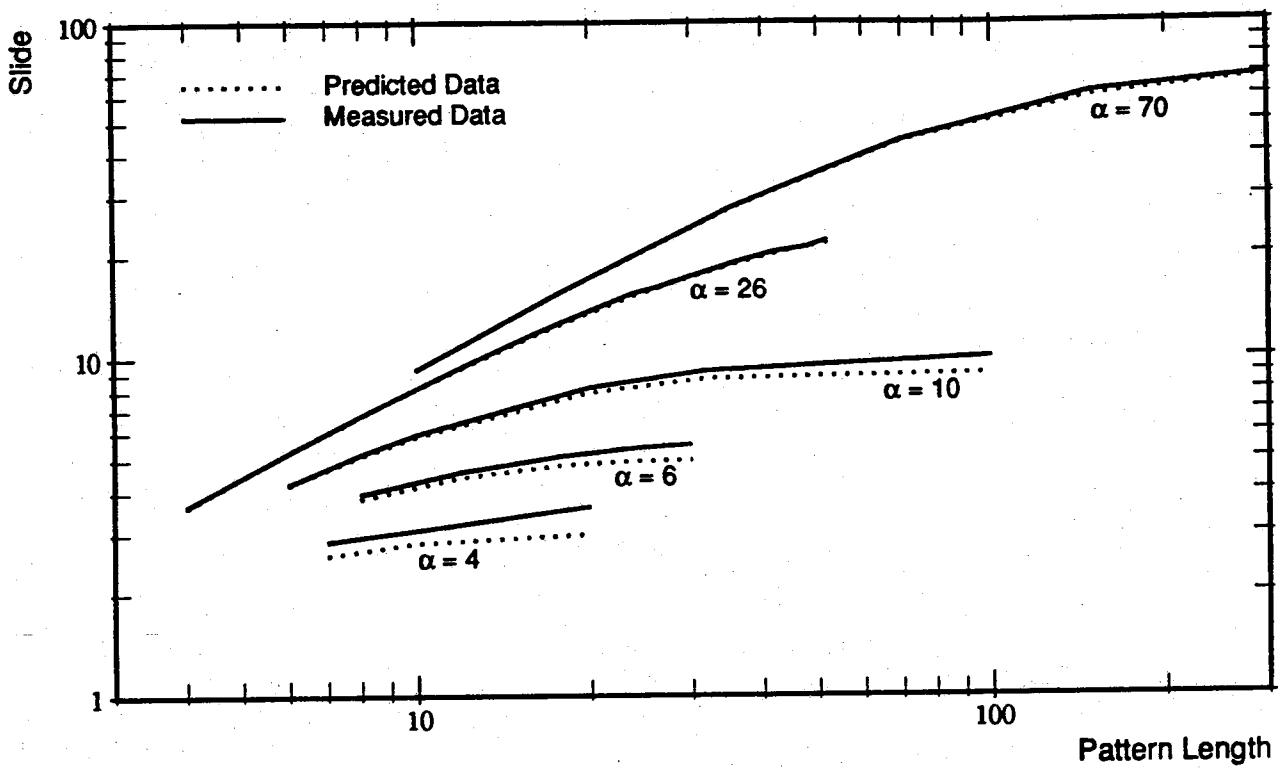

Figure 2: Predicted and measured slide on short patterns with α= (4,6,10,26,70).

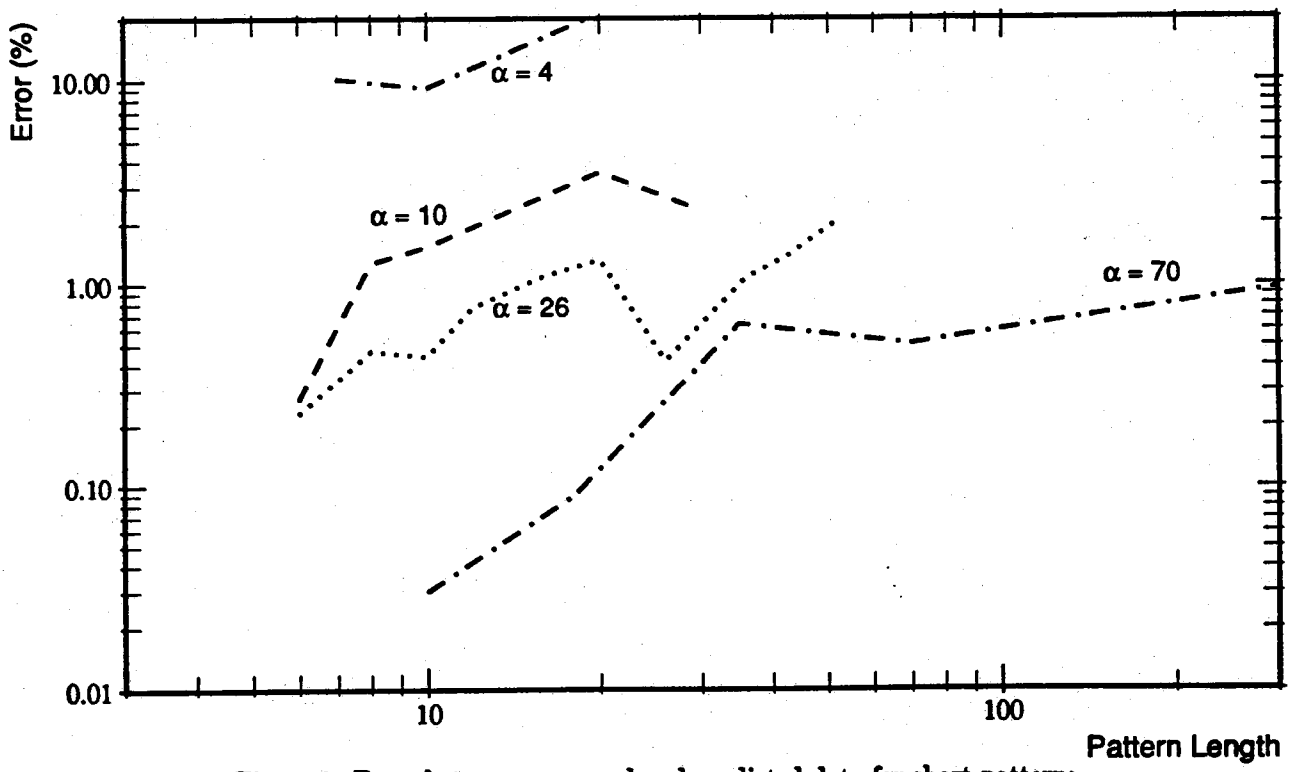

Figure 3: Error between measured and predicted data for short patterns.

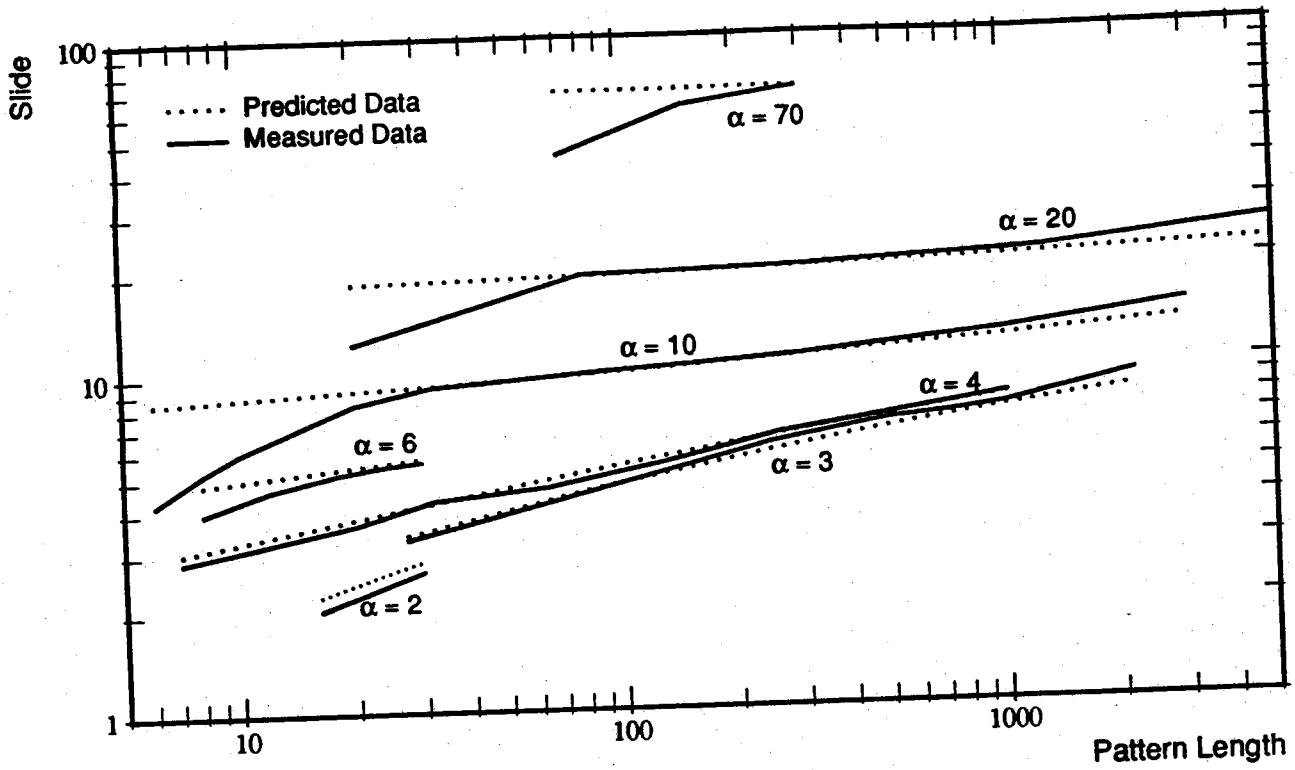

Figure 4: Predicted and measured slide on long patterns with α= (2,3,4,6,10,20,70).

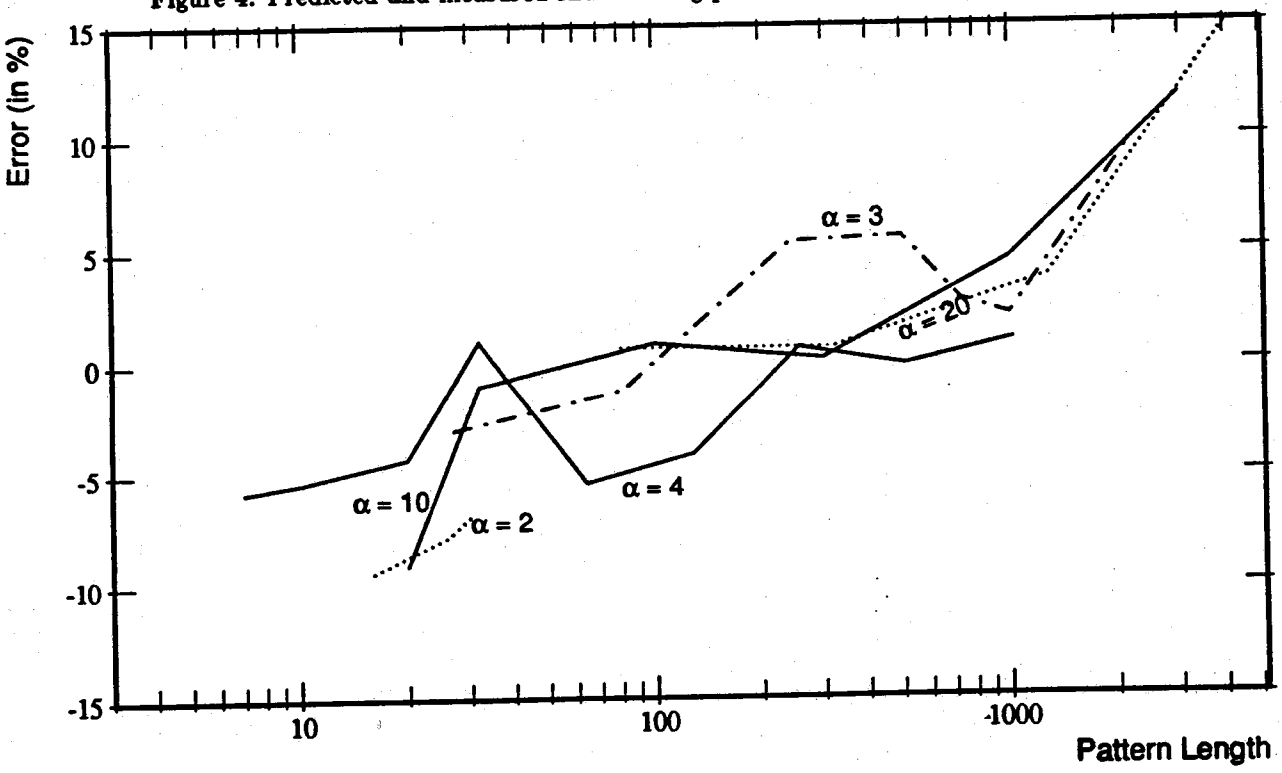

Figure 5: Error between measured and predicted data for long patterns.

Each approximation is accurate over its designated range of M. Figure 6 shows the predicted and measured number of comparisons for $\alpha = 10$ from small to large M values for N= 50,000. The short and long approximations cross over when $M \approx 2\alpha$.

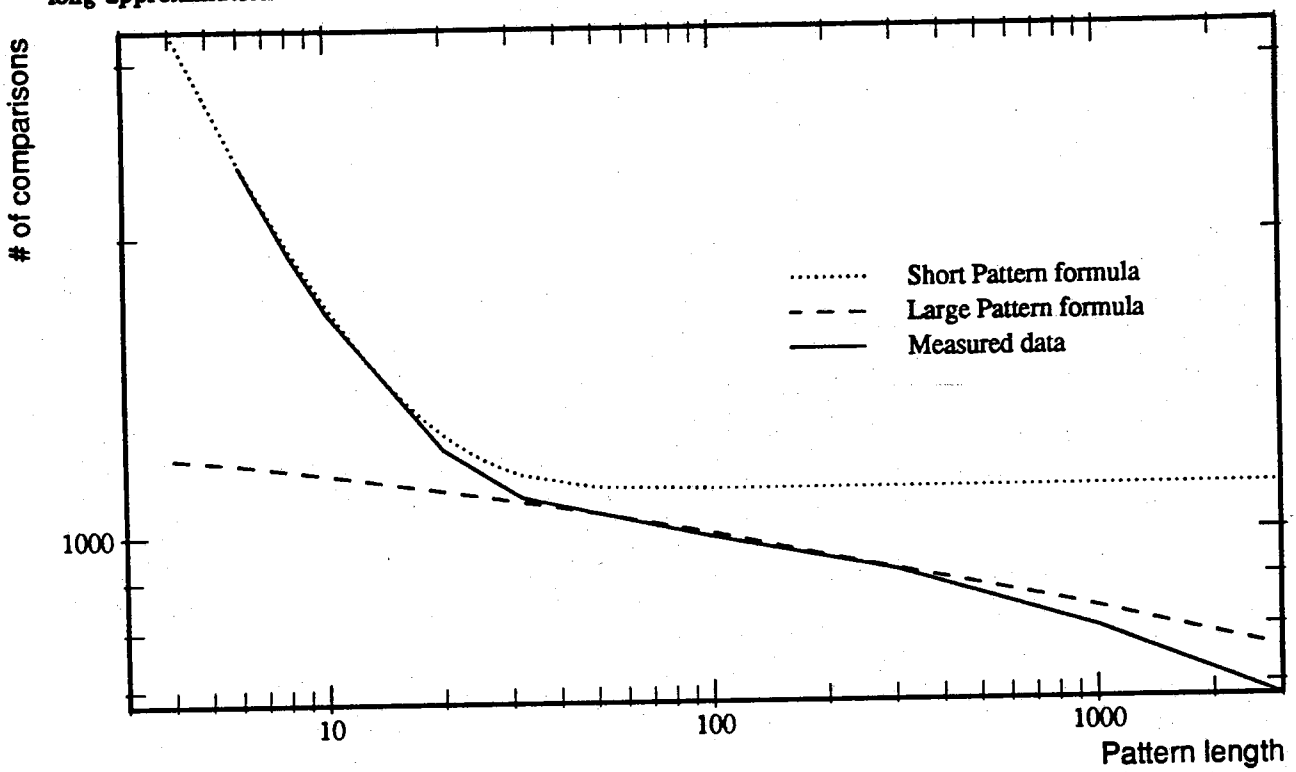

Figure 6: Number of comparisons for $4 \leq M \leq 3000$ with $\alpha = 10$.

Equation 10 starts to break down for very large M, because Simplification 2, which limits the anchor to one index, causes us to underestimate $\Pr(\xi_{a*:m})$ for large m. These states occur rarely, but their expected shift increases as M increases, so that their contribution to $\overline{\text{slide}}$ becomes increasingly significant.

The growing inaccuracy of Equation 10 at very large M is of little practical importance, because a modified algorithm would normally be used. For large M, it is much more efficient to index the Δ_1 table by a block of b characters, rather than a single character [4] [6]. This approach gives an effective alphabet size of α^b which makes Equation 8 applicable. By choosing $b = \lfloor \log_\alpha M \rfloor$, the average number of characters examined is roughly $N \log_\alpha M/M$, which is close to optimal [6].

7 Future Work

Several steps in our derivation could be stronger. In particular, removal of the assumption that the anchor is only indice should yield an expression for $\overline{\Phi(BM)}$ accurate for very large M. We should also correctly

calculate $\overline{s(\xi_{a*:1})}$, rather than using Simplification 4.

References

[1] Alberto Apostolico and Raffaele Giancarlo. The Boyer-Moore-Galil string searching strategies revisited. *SIAM Journal of Computing*, 15(1):98–105, February 1986.

[2] Robert S. Boyer and J. Strother Moore. A fast string searching algorithm. *Communications of the ACM*, 20(10):762–772, October 1977.

[3] Leo J. Guibas and A. M. Odlyzko. A new proof of the linearity of the Boyer-Moore string search algorithm. *SIAM Journal of Computing*, 9:672–682, 1980.

[4] Donald E. Knuth, James H. Morris, and Vaughan R. Pratt. Fast pattern matching in strings. *SIAM Journal of Computing*, 6(2):323–350, June 1977.

[5] R. Schaback. On the expected sublinearity of the Boyer-Moore algorithm. *SIAM Journal of Computing*, 17(4):648–658, August 1988.

[6] Andrew Chi-Chih Yao. The complexity of pattern matching for a random string. *SIAM Journal of Computing*, 8(3):368–387, August 1979.