An Empirical Analysis of Pattern Scan Order in Pattern Matching

M. Oğuzhan Külekci, *Member, IAENG*

***Abstract*— In pattern matching, scanning a given pattern in a particular order greatly influences the performance. This study investigates the effect of different pattern scan orders on natural language text and on DNA sequence data. Besides the well-known right-to-left ordering of Boyer-Moore, and from the least frequent character to most frequent one of Sunday's optimal mismatch algorithm, four alternative character search sequence orderings based on newly introduced *distant n-gram statistics* are proposed within this work. In all experiments, Sunday's pattern matching algorithm, where the characters of a given pattern can be scanned in any order, is used as the main framework. On natural language test data, the alternative pattern scan orders give better results in 60% of the test keywords. On genome data best ordering among the tested six approaches is the right-to-left order.**

***Index Terms*— pattern scan order, distant n-gram statistics, optimal mismatch, pattern matching.**

I. Introduction

Pattern matching is the act of locating exact occurrences of a given pattern in a text. In the last three decades, many algorithms have been proposed to perform this task in an efficient way both in time and space. Some of the surveys on the topic are [1], [2], and [3].

Pattern scan order, which may be defined as the search sequence of pattern characters, greatly influences the performance. Various orderings have been proposed until now, and in general, string matching algorithms may be classified according to their pattern scanning order as; from left-to-right (e.g., Knuth-Morris-Prat [4]), from right-to-left (e.g., Boyer Moore [5]), in a specific order(e.g., optimal mismatch [6] and maximal shift [6]), or in random order (e.g., Harspool [7]) [2]. Best theoretical and practical results have been obtained with searching patterns in a specific order, and from right-to-left scanning respectively [2].

The fact that determines the performance of a pattern matching algorithm is the number of character comparisons carried out while scanning a text for a given pattern. Fast detection of mismatches and performing maximum shifts are the main points considered to lower that cost. This study concentrates on ordering the characters of a pattern such that the number of total comparisons achieved while searching for a pattern in a text is minimal.

As an example, Turkish word *yunus* has 120 (5!) distinct orderings. The search of that pattern on 25MB of Turkish text results in different number of character comparisons with each of those permutations. Some of the sample scan orders with the corresponding number of comparisons achieved on search operation are given in Table I. It is seen that the worst ordering accomplishes approximately 8% more character comparisons then the best ordering among the samples given.

Note that the pattern matching algorithm used in experiments is the Sunday's algorithm, which permits to scan a string in any order. The algorithm calculates the bad character and good suffix shifts of the pattern according to the given order, and performs the search operation appropriately. On all over the tests conducted within this study, Sunday's algorithm is run with different scan orders on test patterns. The implementation of the algorithm in Lecroq's handbook of exact string-matching algorithms [2] is used in the experiments conducted in this paper.

Throughout the study, first formal definition of the optimal mismatch problem is given. Then, approximations for the exact solution and statistics required for these approximations are described. The experiments conducted on natural language text and on DNA sequence data are explained and results are represented next.

Table I. Total number of character comparisons occurred while searching for the word *yunus* (dolphin) in 25MB of Turkish text for some sample scanning orders.

Scan Order	Total # of character comparisons
0 (y) - 1 (u) - 2 (n) - 3 (u) - 4 (s)	4946461
4 (s) - 3 (u) - 2 (n) - 1 (u) - 0 (y)	**4920562**
3 (u) - 1 (u) - 4 (s) - 0 (y) - 2 (n)	5019087
1 (u) - 4 (s) - 0 (y) - 2 (n) - 3 (u)	4836415
2 (n) - 1 (u) - 3 (u) - 0 (y) - 4 (s)	**5280164**

Manuscript received March 20, 2007.

M. O. Külekci was with the National Research Institute of Electronics & Cryptology, Kocaeli - Gebze 41470, TURKEY. He is currently completing his military obligation in Turkish Army Gendarme Headquarter, Ankara – Beştepe, TURKEY (e-mail: kulekci@uekae.tubitak.gov.tr).

II. Optimal Mismatch Problem

Let X = x[0..m-1] be a pattern of length *m*, and Y = y[1..n-1] be a text of length *n*, on which the pattern will be scanned. The scan order of pattern characters is shown by S={s_0, s_1, .., s_{m-1}}, where s_i represents the index of the character that is to be scanned at *i*th position on pattern X. Thus, the first character to be checked on pattern X is x_{s_0}, second is x_{s_1} ,and so on. Note that each s_i is distinct and $0 \le s_i < m$.

It is clear that, for length *m*, there is *m*! such possible orderings. As an example, for X=*abc*, where *m*=3, possible sets of S are {0,1,2}, {0,2,1}, {1,2,0}, {1,0,2}, {2,1,0}, and {2,0,1}. If one decides to use the {1,0,2} order, then first *b* will be checked, followed by the inspection of the characters *a* and *c* on pattern X respectively. The goal is to select the appropriate S that results in minimum number of character comparisons during the search performed on Y.

The main idea is to make most informative check at each step of the scanning process given the previously scanned characters of the pattern. As an example, let's say for the sample pattern *yunus* we begin scanning with the second character *'u'*(x_1). Next, we need to decide which character is most unlikely to occur given '*u*' (x_1). That is to find out *k* which minimizes $P(x_k \mid x_1)$, for *k*={0,2,3,4}. Assuming that this probability is minimal with the selection of *k*=4, *'s'* (x_4) should be scanned after *'u'* (x_1). If we continue one more step, now the question is; given that the second position is *'u'* and the fifth position is *'s'*, which of the remaining characters '*y*' (x_0), *'n'* (x_2), or *'u'* (x_3), is most unlikely. Among the possible values of k={0,2,3}, the one that is minimizing the probability $P(x_k \mid x_1,x_4)$ is the answer. By continuing the same way of processing, a pattern scan order S can be deduced. The overall cost of such a pattern scan order S={s_0, s_1, .., s_{m-1}} is formulated in Equation 1. The global solution of the optimal mismatch problem is to select s_0, s_1, .., s_{m-1} such that this cost is minimum.

$$c(S) = P(x_{s_0}).P(x_{s_1} \mid x_{s_0}).P(x_{s_2} \mid x_{s_0}, x_{s_1}) \ldots \\ \ldots P(x_{s_{m-1}} \mid x_{s_0}, x_{s_1}, ..., x_{s_{m-2}}) \quad (1)$$

The difficulty here is to calculate $P(x_{s_k} \mid x_{s_0},x_{s_1},\ldots x_{s_{k-1}})$ with the classical n-gram statistics. That is because; this probability requires the relationship between the characters that are some distance apart, e.g., $P(x_4 \mid x_1)$ means the probability of seeing x_4 after 3 characters from x_1. In other words, that is the probability of observing pattern *"x_1??x_4"*, given x_1 , and whatever the *'?'* characters are.

Distant n-gram (d-n-gram) statistics are proposed to overcome this problem. The d-n-gram statistics of a language is the statistics extracted from characters that are d distance apart from each other. For example, extracting 2-n-gram (d=2) statistics from a text of Y=y[0..q], is equivalent to classical n-gram probabilities that are calculated from texts y[0,2,4,6,….] and y[1,3,5,7,…]. Similarly, 3-n-grams are the normal n-grams extracted from texts y[0,3,6,…], y[1,4,7,…], and y[2,5,8,…].

In this study 4 approaches, which are explained in the next section, are used to approximate the $P(x_{s_k}|x_{s_0},x_{s_1},\ldots x_{s_{k-1}})$ probability by using d-2-grams, where d=1..20 on natural language test data, and d=1..100 on genome sequence data. The upper limit of the distance d value specifies the maximum length of the patterns that can be solved using these approaches. Although it is possible to extract statistics for higher values of d, it is decided that pattern length of 20 for natural language text and 100 for genome data is enough for the test purposes.

III. Approximation Approaches

Informally, $P(x_{s_k} \mid x_{s_0},x_{s_1},\ldots x_{s_{k-1}})$ means the probability of observing a specific character at a specific position given some number of other pattern characters. Exact calculation of that probability requires cumbersome statistics. As an example, seeing *y* at the first position given that the third is *n* and fifth is *s* requires the counts of *n?s* and also *y?n?s* patterns. (*'?'* is used as the wild character.) The situation is much worse for higher number of given characters. Thus, one needs to approximate this probability in some way to be able to calculate the cost of a pattern scan order S as in Equation 1.

The approaches to approximate $P(x_{s_k}|x_{s_0},x_{s_1},\ldots x_{s_{k-1}})$ used in this study are described below. Note that, whenever required $P(x_{s_i}|x_{s_j})$ is calculated as shown in Equation 2 by d-2-grams, where $d=|s_i - s_j|$, and $P(x_{s_j})$ is the frequency of character x_{s_j}.

$$P(x_{s_i} \mid x_{s_j}) = \frac{|s_i - s_j| \_ 2 \_ gram(x_{s_i}, x_{s_j})}{P(x_{s_j})} \quad (2)$$

A. Markov Approach

This approach considers only the last observed character while calculating the desired probability, instead of taking all previously seen characters into account, as shown in Equation 3. That is to make most promising scan according to the last observation, and thus, named as Markov approach.

$$P(x_{s_k} \mid x_{s_0}, x_{s_1}, ..., x_{s_{k-1}}) \cong P(x_{s_k} \mid x_{s_{k-1}}) \quad (3)$$

B. Pessimistic Approach

While calculating the $P(x_{s_k}|x_{s_0},x_{s_1},\ldots x_{s_{k-1}})$, each of the characters at positions s_0, s_1, .., s_{k-1} defines an individual information on character s_k by the probability of $P(x_{s_k} \mid x_{s_i})$. An overall decision based on these individual effects should be given. If we assume the worst case (thus this approach is named pessimistic), the one with the maximum of those probabilities will occur. Then Equation 4 may be used to calculate $P(x_{s_k}|x_{s_0},x_{s_1},\ldots x_{s_{k-1}})$.

$$P(x_{s_k} \mid x_{s_0}, x_{s_1}, ..., x_{s_{k-1}}) \cong \max(P(x_{s_k} \mid x_{s_0}), P(x_{s_k} \mid x_{s_1}), .. P(x_{s_k} \mid x_{s_{k-1}})) \quad (4)$$

C. Optimistic Approach

As oppose to the idea in pessimistic approach, optimistic way assumes the best case and uses the minimum probability among the possibilities as formulated in Equation 5.

$$P(x_{s_k}|x_{s_0},x_{s_1},...,x_{s_{k-1}}) \cong \min(P(x_{s_k}|x_{s_0}),P(x_{s_k}|x_{s_1}),..P(x_{s_k}|x_{s_{k-1}})) \quad (5)$$

D. Average Approach

Another possibility for approximating the desired probability is to take the average of all the probabilities between the x_{sk} and the previously decided x_{si} characters as shown in Equation 6.

$$P(x_{s_k}|x_{s_0},x_{s_1},...,x_{s_{k-1}}) \cong (\frac{1}{k}).\sum_{i=0}^{i=k-1} P(x_{s_k}|x_{s_i}) \quad (6)$$

IV. Experiments

A. Natural Language and DNA Sequence Test Data

Natural language text used in this study is collected from a Turkish daily newspaper and it is approximately 100MB in size. The d-2-gram statistics, where d=2..20, is extracted from 75MB of the text, and the remaining 25MB is used in the experiments. As the alphabet of a natural language is rather large, the portion of the text used for the statistics extraction is preferred to be more than the half. Randomly chosen distinct 50 words for each length of 2 to 20 are used as test patterns in the experiments. Thus, 19*50=950 keywords are scanned in 25MB Turkish text for test purposes.

The DNA sequence data is the first chromosome of human genome [8] that is approximately of size 270 MB. The distance d in d-2-gram statistics collected on genome data is held between 5 and 100 with steps of five. That is, 5-2-grams, 10-2-grams, and similarly up to 100-2-grams are extracted from first half of the DNA sequence, and the remaining half is used in experiments. For each length of 5, 10, 15, …, and 100, 50 distinct test patterns are selected randomly from the data. Totally, 20*50=1000 random test sequences are collected and searched in the second half of the genome sequence during the experiments conducted.

CMU-Cambridge language modeling toolkit [9] is used in collecting the required D-2-gram statistics on all over the study.

B. Methodology

Each of the test patterns is searched on the related natural language or DNA sequence test data. The algorithm used in scanning process is the Sunday's pattern matching algorithm [6] that can be run with any given order of pattern characters. The implementation is taken from the Lecroq's website of pattern matching algorithms [2].

The sequence of characters that minimizes the cost given in Equation 1 should be found according to all approximation approaches described in section III. With this aim, the trellis graph of each test pattern is created and Viterbi [] algorithm is used in finding the min cost path. A sample trellis graph for the word *yunus* is depicted in Figure 1.

State transition probabilities between the nodes of the graph are set according to the approximation approach being tested. As an example, in Figure 1, if we assume that the path to node *n* (x_2) of level 3 has visited *y* (x_0) and *s* (x_4) in levels 1 and 2, the cost of transition from *n* to node *first u* (x_1) of level 4 is $P(x_1|x_0,x_4,x_2)$. If Markov approach is being tested on the pattern, then this probability is calculated as $P(x_1|x_2)$, or if optimistic approach is under consideration, then it is $\min(P(x_1|x_0), P(x_1|x_4), P(x_1|x_2))$.

It is noteworthy here that the winning path should contain each letter only once. Thus, a control is necessary to avoid ending up with sequences like *(u,u,s,u,s)*, *(y,n,y,u,s)*, and etc… While calculating the cost from a node in level k to a node in level (k+1), it is checked whether the target node has been visited in the history of the source node first. If so, this transition is prohibited.

After the detection of the minimum cost path by Viterbi, the characters of the pattern are ordered according to the winning path. Sunday's algorithm is run with this ordering and the total number of character comparisons is counted.

This process; running Viterbi with the appropriate probabilities of the current approach, ordering the pattern characters according to the min cost path, and running Sunday's algorithm on test data with this ordering, is performed for each of the 4 approaches on each of the test patterns.

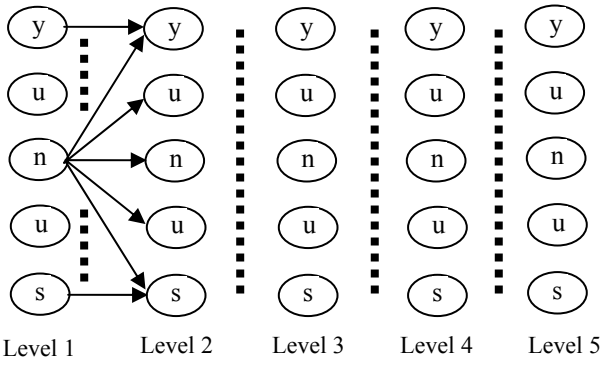

Figure 1. The trellis graph of sample word *yunus*

Besides the four new approaches introduced, Sunday's optimal mismatch (OM), which offers to scan pattern characters from the least frequent one to the most frequent, is also carried out on test patterns. By this way new approaches are benchmarked against the Sunday's OM.

Boyer-Moore type from right-to-left ordering is applied also to be able to observe the overall effect of pattern scan orders. It has been accepted as a standard way of searching in many of the text editors, and thus supposed to be a good base line of experiments throughout this study.

In summary, 950 Turkish keywords and 1000 genome sequences are searched within 25MB of text and 135MB of DNA sequence data respectively. Each pattern is scanned 6

times with pattern scan orders of 4 new approaches, Sunday's OM, and Boyer-Moore style from right-to-left order. Results are discussed in next section.

V. Results

Table II shows the number of times each approach ended up with minimum number of character comparisons on test data. In some cases more than one approach decide on the same pattern scan order that results minimum number of comparisons. In such situations, all these approaches are assumed as winning. Thus, the sums of the columns are more than the exact number of patterns 950 and 1000 respectively.

Sunday's optimal mismatch (OM) order and from right-to-left ordering of Boyer-Moore (BM) give the best results on natural language and genome data respectively.

On natural language text, the pessimistic approach introduced in this study is compatible with OM, where the results of the rest do not make much sense. Although BM order is the worst on natural language text, it is definitely the best on genome data. The statistical approaches are not successful as they are on language text here. Most probably that is because of the randomness in DNA sequences.

Table II. Number of times each approach ended up with minimum number of character comparisons on test data.

Approach	*On Natural Language Text*	*On Genome Data*
BM	31	**391**
OM	**392**	193
Greedy	118	111
Pessimistic	**369**	93
Optimistic	226	132
Average	275	93

Table III and Table IV indicate which ordering results in minimum number of comparisons on both test data grouped by the length of the patterns. On each test pattern, the percentage of difference between the maximum and minimum number of character comparisons achieved by the investigated 6 approaches is calculated with the formula ((max_count – min_count) / max_count)*100. It is observed that on the average the difference between the best and worst ordering cause 9.25% and 29.67% more comparisons on natural language and DNA sequence data respectively. This implies that deciding on the correct pattern scan order significantly affects the performance especially on DNA sequence searching.

Here it must be noted that all of the test approaches are reasonable ones that give close results. If left-to-right ordering is taken into account, the gap between the best and worst counts would be much higher. In this study, left-to-right ordering is neglected as it is not used in practical systems.

Table III. How many times each approach result in minimum number of character comparisons on natural language text.

Pattern Length	*APPROACHES*					
	BM	*OM*	*Markov*	*Pes.*	*Opt.*	*Avg.*
2	22	**50**	24	**50**	**50**	**50**
3	7	**37**	25	34	32	37
4	2	22	12	**25**	31	30
5	0	**20**	9	13	17	14
6	0	**18**	6	17	8	13
7	0	16	5	**17**	6	12
8	0	13	6	**15**	8	9
9	0	17	0	**19**	10	6
10	0	17	2	**19**	4	12
11	0	**20**	5	11	9	8
12	0	**18**	6	12	5	10
13	0	**14**	3	12	6	15
14	0	**18**	5	14	8	8
15	0	**21**	0	19	5	6
16	0	14	1	**21**	6	10
17	0	**20**	4	**20**	5	4
18	0	**20**	1	17	5	9
19	0	**18**	3	15	5	12
20	0	**19**	1	**19**	6	10
Totals	31	392	118	369	226	275

Table IV. How many times each approach result in minimum number of character comparisons on DNA sequence data.

Pattern Length	*APPROACHES*					
	BM	*OM*	*Markov*	*Pes.*	*Opt.*	*Avg.*
5	**14**	8	9	10	12	10
10	**23**	3	8	3	5	8
15	**17**	5	7	9	9	3
20	**20**	9	1	7	8	5
25	**16**	10	4	8	6	6
30	**20**	10	6	4	7	3
35	**23**	9	4	2	7	5
40	**19**	9	4	4	9	5
45	**11**	17	6	5	6	5
50	**14**	15	7	3	6	5
55	**29**	8	5	2	6	0
60	**23**	10	5	4	2	6
65	**21**	12	7	5	1	4
70	**25**	6	6	2	7	4
75	**13**	15	4	5	9	4
80	**16**	10	9	5	4	6
85	**18**	6	7	7	8	4
90	**22**	12	5	1	6	4
95	**26**	5	0	5	11	3
100	**21**	14	7	2	3	3
Total	391	193	111	93	132	93

Although none of the newly introduced approaches are absolute best on both data sets, they gave better results than the classical techniques in approximately 60% of the natural language test patterns and in 40% of the DNA sequence patterns. That indicates there is still room for research on finding the best pattern scan order. It is also noteworthy that if the minimum number of comparisons is achieved by one of the

proposed approaches, the gain is more than 6% on natural language text.

VI. Conclusion

In this study an empirical analysis of the effect of pattern scan order in pattern matching is achieved. Number of character comparisons performed during a search operation is used as the performance metric. Six distinct approaches to order the characters of a given pattern are tested on natural language and DNA sequence data sets. These are from right-to-left scanning (as in Boyer-Moore algorithm), from least frequent to most frequent one scanning (as in Sunday's optimal mismatch algorithm), and four newly proposed ones named Markov, optimistic, pessimistic, and average approaches. Distant n-gram statistics (*d*-*n*-gram), which is the statistics of characters *d* distance apart, is introduced with the proposed techniques.

On natural language text, pessimistic approach and Sunday's optimal mismatch orderings gave compatible results where Sunday's was a little better then the pessimistic ordering. On DNA sequence data, best results have been obtained definitely by Boyer-Moore's from right-to-left ordering.

The effect of pattern scan order is evaluated to be approximately 10% and 30% on natural language and DNA sequence data respectively. In other words, among the tested orderings the number of character comparisons between the worst and best differ 10% and 30% on related test sets. This indicates the importance of pattern scan order especially on DNA sequence searching.

The global solution to find the best pattern scan order of a given pattern that requires minimum number of character comparisons has not been accomplished yet. Thus, methods to establish better scanning orders can be studied in further studies. In addition, given a pattern, techniques that decide which ordering fits best can also be investigated. By this way, before scanning a pattern, ordering of pattern characters can be achieved by applying appropriate ordering methodology.

References

[1] M. Crochemore and T. Lecroq, "Pattern matching and text compression algorithms", in *The Computer Science and Engineering Handbook*, A.B. Tucker, Ed., CRC Press: Boca Raton, 2003, Chapter 8.

[2] C. Charras, and T.Lecroq, *Handbook of exact string matching algorithms*, King's Collage Publications, 2004, Available: http://www-igm.univ-mlr.fr /~lecroq/string/.

[3] A. V. Aho, "Algorithms for finding patterns in strings", in *Handbook of Theoretical Computer Science*, Elsevier, Amsterdam, 1990, vol. A, pp.255-300.

[4] D. E. Knuth, J. H. Morris, and V. R. Pratt, "Fast pattern matching in strings", *SIAM Journal of Computing*, vol. 6, 1977, pp.323-350.

[5] R. S. Boyer and J. S. Moore, "A fast string searching algorithm", *Communications of the ACM*, vol. 20, 1977, pp.762-772.

[6] D. M. Sunday, "A very fast substring search algorithm", *Communications of the ACM*, vol. 33, 1990, pp.132-142.

[7] R. N. Horspool, "Practical fast searching in strings", *Software Practice & Experience*, vol. 10, 1980, pp.501-506.

[8] Human Genome Project Chromosome 1, Project Gutenberg Release #3501, available: http://onlinebooks.library.upenn.edu/webbin/gutbook/lookup?num=3501.

[9] P. Clarkson and R. Rosenfeld, "Statistical language modeling using the CMU-Cambridge toolkit", Proceedings of the EuroSpeech'97, Rhodes, Greece, 1997, pp. 2707-2710.

[10] L.R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition", *Proceedings of the IEEE 77*, vol.2, 1989, pp. 247-286.